孙祥 徐流美 吴清 编著

MATLAB 7.0 基础教程

清华大学出版社

MATLAB 7.0 基础教程

孙 祥 徐流美 吴 清 编著

清华大学出版社

北 京

内 容 简 介

本书结合科学研究和工程中的实际需要，系统地介绍了数学软件 MATLAB 7.0 的基本功能，包括数值计算功能、符号运算功能和图形处理功能等，并在此基础上精心设计了丰富的实例。同时本书还介绍了 MATLAB 7.0 在科学计算中的一些应用。

本书内容由浅入深，适用于 MATLAB 软件的初、中级用户，特别适合作为大学教材，也可以作为科学与工程计算科技人员的学习资料。

图书在版编目(CIP)数据

MATLAB 7.0 基础教程/孙祥等编著. —北京：清华大学出版社，2005.5

ISBN 7-302-10711-4

Ⅰ.M… Ⅱ.孙… Ⅲ.计算机辅助计算—软件包，MATLAB 7.0—教材 Ⅳ.TP391.75

中国版本图书馆 CIP 数据核字(2005)第 023103 号

出 版 者：清华大学出版社　　地　　址：北京清华大学学研大厦
http://www.tup.com.cn　　邮　　编：100084
社 总 机：010-62770175　　客户服务：010-62776969
组稿编辑：胡辰浩　　文稿编辑：鲍　芳
封面设计：信　京　　版式设计：康　博
印 刷 者：北京鑫霸印务有限公司　　装 订 者：北京鑫海金澳胶印有限公司
发 行 者：新华书店总店北京发行所
开　　本：185×260　印张：25　字数：577 千字
版　　次：2005 年 5 月第 1 版　2005 年 5 月第 1 次印刷
书　　号：ISBN 7-302-10711-4/TP・7238
印　　数：1～4000
定　　价：35.00 元

前　　言

MATLAB 源于 Matrix Laboratory 一词，原为矩阵实验室的意思。它的最初版本是一种专门用于矩阵数值计算的软件。随着 MATLAB 的逐步市场化，其功能也越来越强大，特别是本书介绍的 MATLAB 7.0，是一门集数值计算、符号运算和图形处理等多种功能于一体的科学计算软件包。它还包含许多专用工具箱，可以满足不同专业用户的需求。如科学计算、动态仿真、系统控制、数据采集、模糊逻辑、金融财政、图形处理、信号处理、数据统计和器材控制等。

目前，MATLAB 已经得到相当程度的普及，它不仅成为各大公司和科研机构的专用软件，在大学校园也得到了普及，许多本科和专科的学生借助它来学习大学数学和计算方法等课程，而硕士生和博士生在做科学研究时，也经常要用 MATLAB 进行数值计算和图形处理。可以说，MATLAB 软件在大学校园已经有了相当的普及，它已经深入到了各个专业的很多学科。本书主要介绍 MATLAB 7.0 的基本功能，包括 MATLAB 7.0 在数值计算、符号运算和图形处理方面的常用功能。同时还在本书的后几章着重介绍了 MATLAB 7.0 在微积分、拟合、插值和常微分方程等科学计算方面的应用。

全书共分为 17 章，第 1 章是 MATLAB 7.0 简介，介绍 MATLAB 语言的基本情况和优缺点，并特别介绍了 MATLAB 7.0 的最新特点；第 2 章介绍了 MATLAB 7.0 的安装和用户界面，对使用界面的认识是掌握 MATLAB 7.0 的基础；第 3 章介绍了 MATLAB 的基本使用方法，通过对本章的学习，读者可以编写简单的 MATLAB 7.0 程序，逐步领略 MATLAB 7.0 强大的数值计算功能；第 4 章介绍 MATLAB 7.0 的数值计算功能；第 5 章介绍两种特殊的 MATLAB 变量，单元型变量和结构型变量；第 6 章介绍字符串的操作；第 7 章介绍多项式的使用；第 8 章介绍关系和逻辑运算；第 9 章介绍 MATLAB 7.0 在符号运算方面的功能；第 10 章介绍函数的 M 文件，该章是编写 MATLAB 程序最重要的内容；第 11 章介绍了文件和数据的导入和导出；第 12 章介绍 MATLAB 7.0 的图形处理功能，从而可以使自己的成果可视化；第 13 章介绍 MATLAB 7.0 的句柄图形，这是进行 GUI 设计的基础；第 14 章介绍图形界面 GUI；第 15 章至第 17 章介绍 MATLAB 7.0 在科学计算方面的一些专题，包括微积分、拟合和插值、普通方程以及常微分方程等。本书可以满足不同类型读者的需求，具有很强的实用性，这是本书的特点之一。

本书是很多人智慧的结晶，由孙祥、徐流美和吴清主编，由徐道磊和邓晓蔚主审，此外，参加本书编写的人员还有李宁波、吴洁、宁宇、倪停、谌凤萍、王喜红、曹向东、徐迎春、戴海霞、周应来、胡小艳、李博、张志广、祈文睿、幸洪福、梁克红、杨爱军、袁任阁和李明耿等。本书在编写过程中参考了一些资料，对这些资料的作者深表感谢。由于作者水平有限，书中难免有不足之处，欢迎广大读者批评指正。

编　者

目　　录

第 1 章　MATLAB 7.0 简介 ······ 1

1.1　MATLAB 简介 ······ 1

1.1.1　MATLAB 的初步知识 ······ 1

1.1.2　MATLAB 的优点 ······ 2

1.1.3　MATLAB 的缺点 ······ 3

1.2　MATLAB 7.0 的新特点 ······ 4

1.3　获取 MATLAB 7.0 最新信息的途径 ······ 5

1.4　习题 ······ 6

第 2 章　MATLAB 7.0 的安装和用户界面 ······ 7

2.1　MATLAB 7.0 的安装 ······ 7

2.2　MATLAB 7.0 用户界面概述 ······ 11

2.2.1　启动 MATLAB 7.0 ······ 11

2.2.2　MATLAB 7.0 的主菜单 ······ 12

2.2.3　MATLAB 7.0 的工具栏 ······ 13

2.2.4　MATLAB 7.0 的窗口 ······ 13

2.3　MATLAB 7.0 的路径搜索 ······ 16

2.3.1　MATLAB 7.0 的当前目录 ······ 16

2.3.2　MATLAB 7.0 的路径搜索 ······ 16

2.4　MATLAB 7.0 帮助系统的使用 ······ 18

2.4.1　帮助窗口 ······ 18

2.4.2　命令窗口查询帮助 ······ 23

2.5　习题 ······ 25

第 3 章　基本使用方法 ······ 26

3.1　简单的数学运算 ······ 26

3.2　MATLAB 7.0 的数据类型 ······ 30

3.2.1　常量和变量 ······ 31

3.2.2　浮点数和复数 ······ 34

3.3　习题 ······ 36

第 4 章　数值计算功能 ······ 37

4.1　向量及其运算 ······ 37

4.1.1 向量的生成……37
4.1.2 向量的基本运算……38
4.2 矩阵及其运算……41
4.2.1 矩阵的生成……42
4.2.2 矩阵的基本数值运算……42
4.2.3 矩阵的特征参数运算……46
4.2.4 矩阵的分解运算……54
4.2.5 矩阵的一些特殊处理函数……61
4.2.6 特殊矩阵的生成……62
4.3 数组及其运算……68
4.3.1 数组寻址和排序……68
4.3.2 数组的基本数值运算……70
4.3.3 数组的关系运算……72
4.3.4 数组的逻辑运算……74
4.4 稀疏型矩阵……74
4.4.1 稀疏矩阵的生成……74
4.4.2 稀疏矩阵与满矩阵的相互转换……76
4.4.3 稀疏矩阵的操作……80
4.5 习题……83
第 5 章 单元数组和结构……85
5.1 单元数组……85
5.1.1 单元数组的生成……85
5.1.2 单元数组的操作……86
5.2 结构型变量……90
5.3 习题……96
第 6 章 字符串……98
6.1 设定字符串……98
6.2 字符串的操作……99
6.2.1 字符串元素的读取……99
6.2.2 字符串的基本变换……100
6.2.3 字符串的运算……103
6.3 习题……112
第 7 章 多项式……113
7.1 多项式的创建……113
7.1.1 直接输入系数向量创建多项式……113
7.1.2 特征多项式输入法……113

7.1.3 由多项式的根逆推多项式 …… 114
7.2 多项式的运算 …… 114
7.2.1 多项式的求值 …… 115
7.2.2 求多项式的根 …… 116
7.2.3 多项式的四则运算 …… 116
7.3 习题 …… 119
第8章 关系和逻辑运算 …… 120
8.1 关系操作符 …… 120
8.2 逻辑操作符 …… 121
8.3 关系与逻辑函数 …… 122
8.4 NaNs 和空矩阵 …… 123
8.4.1 NaNs 的处理 …… 124
8.4.2 空矩阵的处理 …… 125
8.5 各种运算符的优先级 …… 126
8.6 习题 …… 127
第9章 符号运算 …… 129
9.1 符号变量的生成和使用 …… 129
9.1.1 符号变量、符号表达式和符号方程的生成 …… 129
9.1.2 符号变量的基本操作 …… 131
9.1.3 符号表达式(符号函数)的操作 …… 135
9.2 符号矩阵的生成和运算 …… 142
9.2.1 符号矩阵的生成 …… 142
9.2.2 符号矩阵及符号数组的运算 …… 145
9.3 符号微积分 …… 151
9.3.1 符号极限 …… 152
9.3.2 符号微分和求导 …… 152
9.3.3 符号积分 …… 154
9.4 符号积分变换 …… 155
9.4.1 Fourier 变换及其逆变换 …… 155
9.4.2 Laplace 变换及其逆变换 …… 156
9.4.3 Z 变换及其反变换 …… 158
9.5 符号代数方程的求解 …… 159
9.5.1 符号线性方程组的求解 …… 159
9.5.2 符号非线性方程组的求解 …… 160
9.5.3 一般符号代数方程组的求解 …… 161
9.6 符号微分方程的求解 …… 163

9.7 图示化符号函数计算器······165
9.7.1 单变量符号函数计算器······165
9.7.2 泰勒级数逼近计算器······168
9.8 利用 maple 的深层符号计算资源······169
9.8.1 maple 命令的调用······169
9.8.2 mfun 命令的使用······171
9.8.3 maple 库函数在线帮助的检索树······171
9.9 习题······173
第 10 章 MATLAB 7.0 程序设计······175
10.1 M 文件入门······175
10.1.1 M 文件的基本特点······175
10.1.2 脚本式 M 文件······177
10.1.3 函数式 M 文件······180
10.2 MATLAB 7.0 程序控制······183
10.2.1 顺序结构······184
10.2.2 选择语句······184
10.2.3 分支语句······188
10.2.4 模块······189
10.2.5 for 循环语句······190
10.2.6 while 循环语句······193
10.2.7 人机交互命令······194
10.3 变量和函数种类······199
10.3.1 函数变量及其作用域······199
10.3.2 函数的分类······203
10.3.3 函数句柄······206
10.4 程序设计的辅助函数······209
10.4.1 执行函数······209
10.4.2 容错函数······211
10.4.3 时间运算函数······213
10.5 程序的调试和优化······221
10.5.1 程序的调试······221
10.5.2 程序的优化······227
10.6 M 文件举例······232
10.7 习题······236
第 11 章 文件和数据的导入与导出······238
11.1 本机数据文件······238

11.1.1 文件的存储······238
11.1.2 文件的打开······239
11.2 数据导入和导出······241
11.3 低级文件 I/O······242
11.4 习题······242

第 12 章 图形处理······244
12.1 基本的绘图命令······244
12.1.1 图形窗口简介······244
12.1.2 基本的绘图操作······245
12.1.3 图形注释······260
12.1.4 特殊图形的绘制······279
12.2 交互式绘图操作······294
12.3 图形的高级控制······298
12.3.1 视点控制和图形的旋转······298
12.3.2 颜色的使用······300
12.3.3 光照控制······304
12.4 习题······306

第 13 章 句柄图形······307
13.1 句柄图形对象······307
13.2 通用函数 get 和 set······315
13.2.1 get 函数······316
13.2.2 set 函数······317
13.3 查找对象······321
13.4 堆积次序······324
13.5 默认属性······324
13.6 习题······325

第 14 章 创建图形用户界面 GUI······327
14.1 GUI 对象层次结构······327
14.2 GUI 的基本知识······328
14.2.1 启动 GUI······328
14.2.2 布局(Layout)编辑器······329
14.2.3 GUIDE 模板介绍······330
14.2.4 运行 GUI······330
14.3 创建 GUI 对象······331
14.3.1 GUI 窗口的布局······331
14.3.2 GUI 控件的属性控制······334

14.3.3　菜单的添加 ········ 335
14.4　GUI 编程 ········ 340
14.4.1　GUI 的 M 文件 ········ 340
14.4.2　给 GUI 的控件响应编制程序 ········ 342
14.4.3　使用句柄结构进行 GUI 数据操作 ········ 346
14.5　习题 ········ 348
第 15 章　微分和积分 ········ 350
15.1　数值微分 ········ 350
15.1.1　使用 diff 函数求数值微分 ········ 350
15.1.2　使用 gradient 函数求近似梯度 ········ 351
15.1.3　jacobian 函数求多元函数的导数 ········ 352
15.2　函数的数值积分 ········ 353
15.2.1　一元函数的数值积分 ········ 353
15.2.2　二元及三元函数的数值积分 ········ 357
15.3　习题 ········ 360
第 16 章　拟合和插值 ········ 361
16.1　最小二乘法实现曲线拟合 ········ 361
16.2　曲线插值 ········ 365
16.2.1　拉格朗日插值 ········ 365
16.2.2　hermite 插值 ········ 367
16.2.3　三次样条插值 ········ 369
16.3　习题 ········ 373
第 17 章　普通方程和微分方程 ········ 374
17.1　方程组的求解 ········ 374
17.1.1　线性方程组的解法 ········ 374
17.1.2　非线性方程组的解法 ········ 381
17.2　微分方程的求解 ········ 382
17.2.1　常微分方程的数值求解 ········ 382
17.2.2　偏微分方程的数值求解 ········ 385
17.3　习题 ········ 386

第1章　MATLAB 7.0简介

MATLAB 是一种功能十分强大，运算效率很高的数字工具软件，全称是 Matrix Laboratory。起初它是一种专门用于矩阵运算的软件，经过多年的发展，MATLAB 已经发展成为一种功能强大的软件，几乎可以解决科学计算中的任何问题。总之，矩阵和数组是 MATLAB 的核心，因为 MATLAB 中的所有数据都是以数组来表示和存储的。除了常用的矩阵代数运算值外，MATLAB 还提供了非常广泛和灵活的方式处理数据集的数组运算功能。另外，MATLAB 除了对矩阵提供了强大的处理能力之外，还具有一种与其他高级语言相似的编程特性。同时它还可以与 Fortran 和 C 语言混合编程，进一步扩展了其功能。在图形可视化方面，MATLAB 提供了图形用户界面(GUI)，使得用户可以进行可视化编程。因此，MATLAB 就把数据结构、编程特性以及图形用户界面完美地结合到一起。

本章主要介绍 MATLAB 的一些基本情况，让大家对该软件有一个整体的认识。它主要包括 MATLAB 的功能、发展历史以及 MATLAB 7.0 的新特点等，由于 MATLAB 软件在不断地更新，所以，也要介绍获取 MATLAB 7.0 最新信息的途径。

1.1　MATLAB 简介

MATLAB 最初是由 Cleve Moler 用 Fortran 语言设计的，有关矩阵的算法来自 Linpack 和 Eispack 课题的研究成果；现在的 MATLAB 程序是 MathWorks 公司用 C 语言开发的。本节主要介绍 MATLAB 的整体情况及其特点。

1.1.1　MATLAB 的初步知识

起初，MATLAB 是专门用于矩阵计算的一种数学软件，但伴随着 MATLAB 的逐步市场化，它的功能也越来越强大，从 MATLAB 4.1 开始，MATLAB 开始拥有自己的符号运算功能，从而使 MATLAB 可以替代其他一些专用的符号计算软件。

在 MATLAB 环境下，用户可以集成地进行程序设计、数值计算、图形绘制、输入输出、文件管理等多项操作。在美国的一些大学里，MATLAB 已经成为对数值线性代数以及其他一些高等应用数学课程进行辅助教学的有益工具。在工程技术界，MATLAB 也被用来解决一些实际课题和数学模型问题。典型的应用包括数值计算、算法预设计与验证，以及一些特殊的矩阵计算应用，如自动控制理论、统计和数字信号处理(时间序列分拆)等。

MATLAB 是一个很大的软件，有着非常强大的功能，仅是基本的 MATLAB 产品就有 1000 个以上的内部函数可供调用，这比其他任何工具提供的函数都要多。而且，由于

MATLAB 具有良好的开放性，它与符号运算功能最强大的工具软件 Maple 之间也有接口。这样，MATLAB 在数值计算、符号运算和图形处理等方面在同类产品中占据了优势，可以说，由于 MATLAB 的强大功能，再加上它本身比较简单易学，MATLAB 已成为高校师生、科研人员和工程技术人员的最佳选择。掌握 MATLAB 将给您的工作和学习带来巨大的便捷，可以极大地提高工作效率和质量。

1.1.2 MATLAB 的优点

与其他的计算机高级语言相比，MATLAB 有着许多非常明显的优点，介绍如下：

1. 容易使用

MATLAB 允许用户以数学形式的语言编写程序，用户在命令窗口中输入命令即可直接得出结果，这比 C、Fortran 和 Basic 等高级语言都要方便得多。由于它是用 C 语言开发的，它的流程控制语句与 C 语言中的相应语句几乎一致。所以，初学者只要有 C 语言的基础，就会很容易掌握 MATLAB 语言。

2. 可以由多种操作系统支持

MATLAB 支持多种计算机操作系统，比如由 windous95/98/2000/XP 以及许多不同版本的 UNIX 操作系统提供支持。而且，在一种操作系统下编制的程序转移到其他的操作系统下时，程序不需要作出任何修改。同样，在一种平台上编写的数据文件转移到另外的平台时，也不需要作出任何修改。因此，用户编写的 MATLAB 程序可以自由地在不同的平台之间转移。这给用户带来了很大的方便。

3. 丰富的内部函数

MATLAB 的内部函数库提供了相当丰富的函数，这些函数可以解决许多基本问题，如矩阵的输入。在其他语言中(比如 C 语言)，要输入一矩阵，先要编写一个矩阵的子函数，而 MATLAB 语言则提供了一个人机交互的数学系统环境，该系统的基本数据结构是矩阵，在生成矩阵对象时，不要求作明确的维数说明。与利用 C 语言或 Fortran 语言编写数值计算的程序设计相比，利用 MATLAB 可以节省大量的编程时间。这给用户节省很多的时间，使用户能够把自己的精力放在创造方面，而把繁琐的问题交给内部函数来解决。

除了这些数量巨大的基本内部函数外，MATLAB 还有为数不少的工具箱。这些工具箱用于解决某些特定领域的复杂问题，比如，使用 Wavelet Toolbox 进行小波理论分析，或者使用 Financial Toolbox 来进行金融方面的问题的研究。同时，用户可以通过网络获取更多的 MATLAB 程序。

4. 强大的图形和符号功能

MATLAB 具有强大的图形处理功能，它本身带有许多绘图的库函数，可以很轻松地画出各种复杂的二维和多维图形。这些图形可以在与运行该程序的计算机连接的任何打印

设备上打印出来，这使得 MATLAB 成为使技术数据可视化的杰出代表。MATLAB 7.0 所绘制的三维图，如图 1-1 所示。

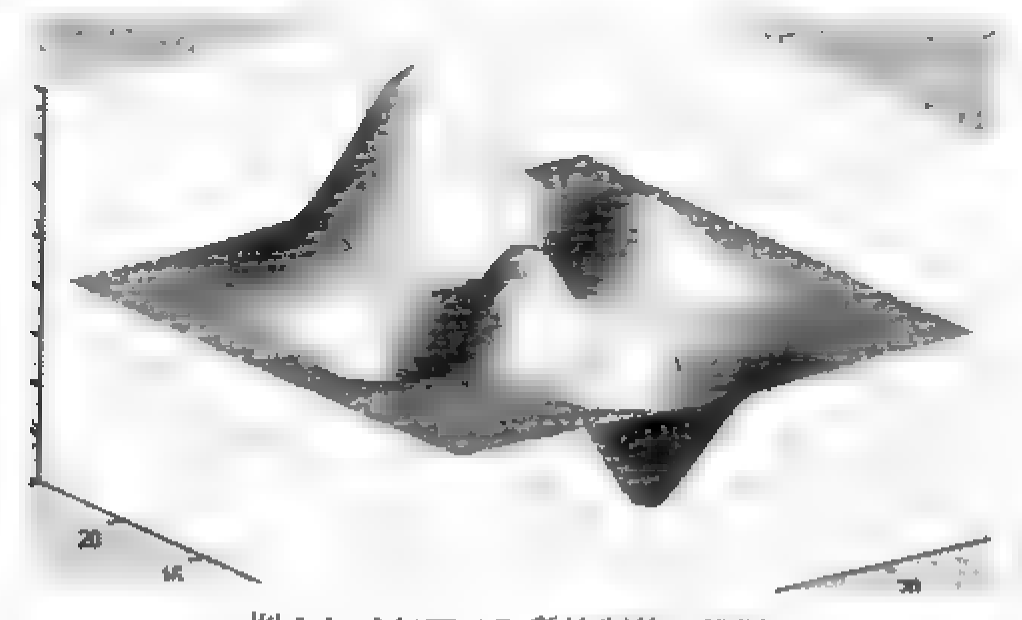


图 1-1　MATLAB 所绘制的三维图

MATLAB 也开发了自己的符号运算功能，特别是 MATLAB 7.0 在这方面的功能丝毫也不逊色于其他的相关软件，如 Mathematic 和 Mathcad 等。因此，用户只需掌握 MATLAB 7.0 这一门语言，就几乎可以解决学习和科研中的所有问题，不必再专门学习一门符号运算语言。同时由于有了 Maple 和 MATLAB 之间的接口，这个问题得到了更好的解决。

5. 可以自动选择算法

在使用其他语言编制程序时，往往会在算法的选择上费一番周折，但在 MATLAB 里，这个问题不复存在。MATLAB 的许多功能函数都带有算法的自适应能力，它会根据情况自行选择最合适的算法，这样，当使用其他程序时，因算法选择不当而引起的譬如死循环等错误，在使用 MATLAB 时可以在很大程度上避免。

6. 与其他软件和语言有良好的对接性

除了上面所提的 MATLAB 与 Maple 的连接外，MATLAB 与 Fortran、C 和 Basic 之间都可以实现很方便的连接，用户只需将已有的 EXE 文件转换成 MEX 文件即可。可见，尽管 MATLAB 除自身已经具有十分强大的功能之外，它还可以与其他程序和软件实现很好的交流，这样可以最大限度地利用各种资源的优势，从而使 MATLAB 编制的程序能够做到最大程度的优化。

1.1.3　MATLAB 的缺点

MATLAB 的缺点主要体现在两个方面。

首先，由于 MATLAB 是一种合成语言，因此，与一般的高级语言相比，用 MATLAB 编写的程序运行起来时间往往要长一些。当然，随着计算机运行速度的不断提高，这个缺点正在逐渐弱化。而且，由于用户在使用 MATLAB 编写程序时比较节省时间，就从编写程序到运行完程序的总的时间来说，使用 MATLAB 仍然比使用其他语言节省时间。

其次，虽然 MATLAB 这套软件比较贵，一般的用户可能支付不起它的高昂费用。但是，由于 MATLAB 具有极高的编程效率，因此，购买 MATLAB 的昂贵费用在很大程度上可以由使用它所编写的程序的价值抵消。所以，就性价比来说，MATLAB 绝对是物有所值。即使是这样，MATLAB 对于一般的用户来说，仍然显得过于昂贵。幸运的是，MATLAB 的开发公司还发行了一种比较便宜的 MATLAB 学生版，这对广大想学习和运用 MATLAB 的用户来说，无疑是一个极好的消息。MATLAB 学生版与 MATLAB 的基本版本几乎一样，可以解决很多科研和学习中遇到的问题。

总而言之，相对于 MATLAB 的优点来说，它的缺点是微不足道的，而且，随着 MATLAB 版本的不断升级，它的缺点已经变得越来越不明显。掌握 MATLAB，必将给我们的学习、科研和工作带来极大的帮助。

1.2 MATLAB 7.0 的新特点

MATLAB 7.0 推出至今已有一段时间了，与早期的 MATLAB 版本相比，使用上具备更多人性化的设计，可以让初学者快速上手。MATLAB 7.0 与 MATLAB 6.5 版本相比，具有以下一些新的特点。

(1) 桌面工具和开发环境的新特点

最根本的变化是 terminal 函数被移开，由 License 命令返回的数据根据字母表的顺序排序，并且只使用小写字母。

在命令窗口环境中，不再支持圆括号匹配。

在帮助窗口环境中，web 函数不再打开帮助系统中的默认页面 URL，而是打开 MATLAB 网络浏览器中的页面。用户可以使用 helpbrowser 选项来打开帮助浏览器中的页面。此外，MATLAB 7.0 版本对常用的网站也进行了更新。

(2) 文件操作、工作间的管理和路径设置

在文件操作、工作间的管理和路径设置 3 个方面，MATLAB 7.0 版本也有了一些改进，MATLAB 7.0 不再将内部函数与搜索路径上的其他 M 文件相区分，而是首先将一个给定的字符以变量搜寻，然后做为当前路径下的 M 文件搜寻，最后才做为搜索路径上的 M 文件搜寻，而低版本的 MATLAB 在将给定字符以变量搜寻之后就以内置函数进行搜寻。所以，如果用户定义了与 MATLAB 内置函数一样的函数，那么该函数可能将替代内置函数进行运行，而在低版本的 MATLAB 中这种情况是不会出现的，如果用户想运行内置函数，可以改变自设函数的名称或是改变搜索路径。

此外，使用 savepath 函数替代了 path2rc 函数，这两个函数功能相同，但是前者更加直观。

(3) 程序的编辑和调试

由于使用了新的注释符号，如果用户编制的 M 文件有一些行只有“%{”和“%}”，这些符号将有可能被当作注释语句的开始和结尾，从而引起语法错误。

在 MATLAB 7.0 版本中，dbstack 函数也得以升级，可以支持嵌套式函数，如果用户在 M 文件中使用了 dbstack 函数，当用户运行 dbstack 函数并将结果返回给某个结构，将得到 3 个域，而低版本的 MATLAB 语言只返回两个域，这 3 个域分别为：

- File：函数出现的文件
- Name：文件中函数的名字
- Line：函数中的线编码

此外，在科学计算，图形绘制以及与其他语言的接口方面，MATLAB 7.0 与以前的版本相比都有了不少改进，详细内容用户可以参见 MATLAB 7.0 的联机帮助，在此不再赘述。

1.3 获取 MATLAB 7.0 最新信息的途径

MATLAB 语言是当今国际上科学界（尤其是自动控制领域）最具影响力、也最有活力的软件。它起源于矩阵运算，并已经发展成一种高度集成的计算机语言。它提供了强大的科学运算、灵活的程序设计流程、高质量的图形可视化与界面设计、便捷的与其他程序和语言接口功能。MATLAB 语言在各国高校与研究单位起着重大的作用。在 Internet 迅速发展的今天，网络成为人们获取信息的最佳途径，而 Mathworks 公司更是特别注意网络在知识传播方面的巨大作用，它的最新资料都会及时地在相关网站上发布。而且 Mathworks 公司和用户之间有着良好的交互性，用户在使用中有什么需求或疑问可以直接通过 EMAIL 与 Mathworks 公司联系。

Mathworks 公司的网址如下。

- www 网址：http://www.mathworks.com
- 匿名 FTP 服务：ftp. Mathworks.com
- ftp. Mathworks.com 的影像站点：Novell.felk.cvut.cz
- 新闻组：comp.soft-sys.matlab
- www 及 ftp 的 Internet IP 地址：144.212.100.10

Mathworks 公司的技术服务联系方式如下。

- 技术支持：support@mathworks.com
- BUG 报道：bugs@mathworks.com
- 文档报道：doc@mathworks.com
- 升级建议：suggest@mathworks.com
- 定购信息：service@mathworks.com
- 订户信息：subscriber@mathworks.com
- 一般信息：info@mathworks.com
- PC 以及 MAK 的升级信息：micro-updates@mathworks.com
- 文件库：matlib@mathworks.com
- MATLAB 文摘：digest@mathworks.com

- FTP 站点：ftpadmin@mathworks.com
- 网络主管：webmaster@mathworks.com

此外，在国内还有许多关于 MATLAB 应用与学习的网站，读者也可以从中获取大量有益的信息，比较有名的如下。

- 清华大学水木清华 bbs 站点：bbs.tsinghua.edu.cn
- 北京大学北大未名 bbs 站点：bbs.pku.edu.cn
- 中国仿真互动论坛：http://www.simwe.com
- 东北大学薛定宇教授维护的 MATLAB 大观园站点：http://www.matlab-world.com

1.4 习　　题

1. 简述 MATLAB 的发展历史及其优缺点。
2. 使用不同的操作系统安装 MATLAB 软件，看看运行 MATLAB 软件时有什么不同。
3. 根据 1.3 节提供的信息，利用互联网了解 MATLAB 语言发展的最新情况。

第2章　MATLAB 7.0的安装和用户界面

本章主要介绍 MATLAB 7.0 的安装和用户界面，通过对本章的学习，用户将学会 MATLAB 软件的安装过程并对用户界面有一个直观的认识。

2.1　MATLAB 7.0 的安装

用户在购买到正版 MATLAB 7.0 后，可以按照相关的说明进行安装，安装过程相对比较简单。安装 MATLAB 7.0 必须具有由 Mathworks 公司提供的合法个人使用许可，如果没有使用许可，用户将无法安装 MATLAB。下面将一步一步指导读者安装 MATLAB 7.0。

(1) 把 MATLAB 7.0 的安装光盘放入光驱中，如果用户没有安装 MATLAB 的其他版本，系统会自动搜索到 autorun 文件并进入安装界面；如果用户已经安装有较低版本的 MATLAB 软件，这时系统会默认已经安装 MATLAB 7.0，界面会一闪而过，此时需要用户自己执行 setup.exe 文件来启动 MATLAB 7.0 的安装程序。安装界面如图 2-1 所示。

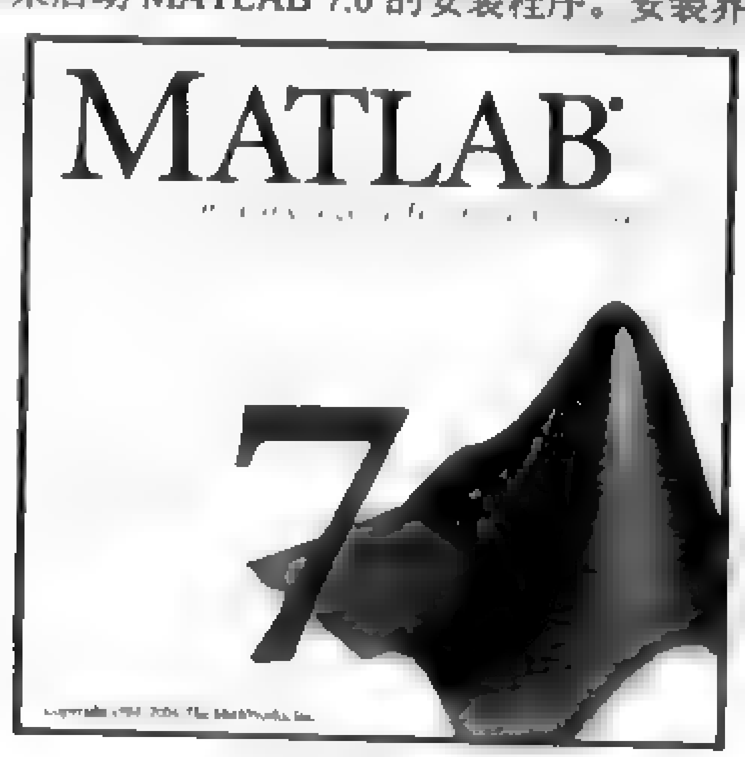


图 2-1　安装界面

(2) 接下来系统会自动弹出 MATLAB 7.0 的欢迎对话框，该对话框的上边有两个单选按钮，选择 Intall 按钮将安装 MATLAB 7.0，而选择另外一个单选按钮将对已经安装的 MATLAB 7.0 的注册码进行更新。这里选择 Install 按钮，如图 2-2 所示。

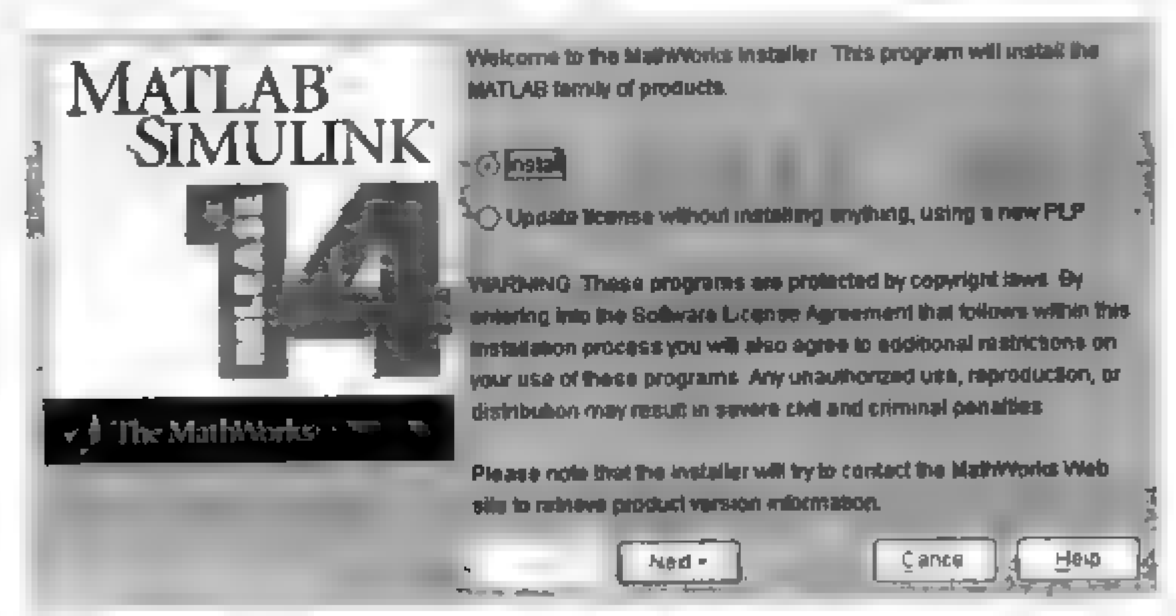


图 2-2　欢迎对话框

(3) 单击欢迎对话框的 Next 按钮，进入安装注册对话框，该对话框中有 3 个文本框需要用户填写，用户在上侧的两个文本框中分别填写自己的姓名和所在的公司，如图 2-3 所示。

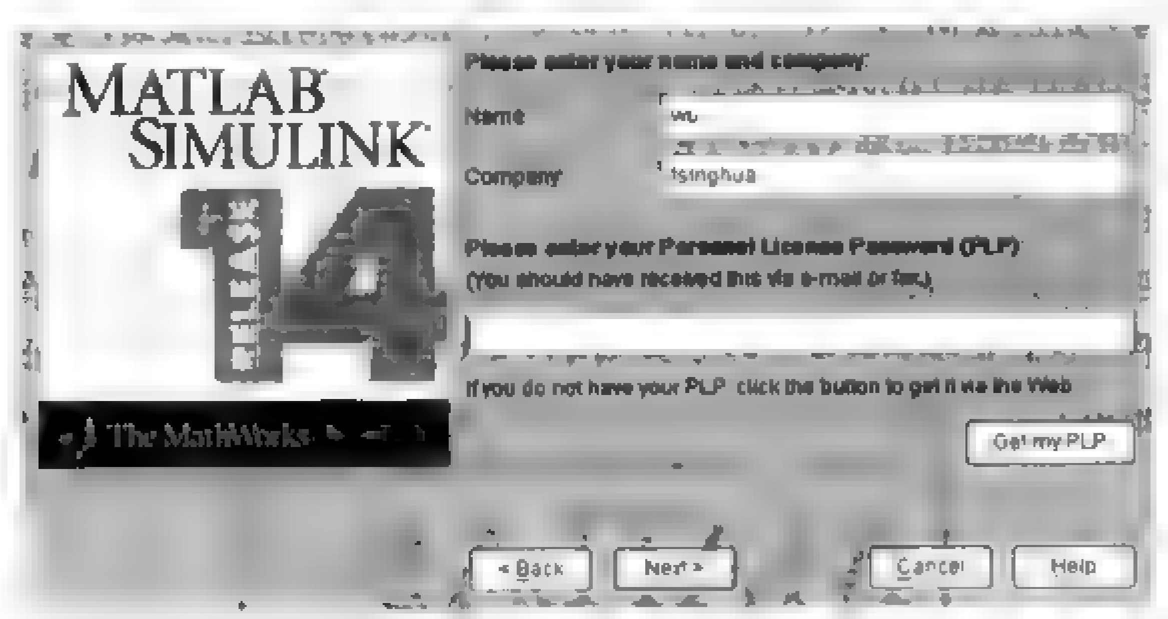


图 2-3　安装注册对话框

(4) 用户在安装注册对话框的文本框中输入所购买的 MATLAB 7.0 软件的注册码，然后单击对话框的 Next 按钮，此时会弹出安装注册协议对话框，如图 2-4 所示。

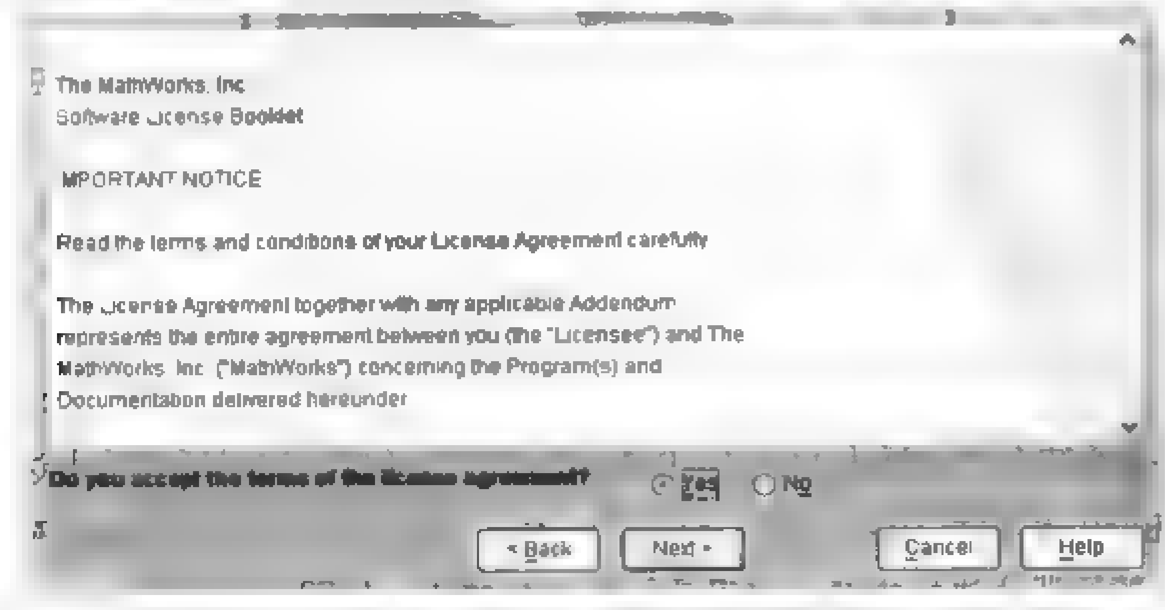


图 2-4　安装注册协议对话框

(5) 用户阅读完对话框中的协议后，如果同意其具体的要求，就可以单击安装注册协议对话框的 Yes 按钮，进入下一步操作。单击安装注册协议对话框上的 Next 按钮，弹出如图 2-5 所示的 MATLAB 7.0 组件选择对话框。用户如果选中 Typical 单选按钮，将安装用户购买的全部 MATLAB 7.0 组件；选中 Custom 单选按钮的话，则将弹出一个文本框用来显示用户所要安装的 MATLAB 7.0 的组件，用户可以从中选择所要需要的组件进行安装。完全安装 MATLAB 7.0 需要约 1.9G 的空间，对于一般用户来说，许多工具箱软件包并不经常用到，而安装过多的工具箱会影响运行速度，因此有选择地安装工具箱就成为了初学者需要注意的问题。在此，选择为默认形式。一般来说，初学者只需选择 MATLAB、Symbolic Math Toolbox 和 MATLAB Complier 工具箱即可以完成基本的编程运算，而只要再装上 MATHLAB C/C++ Graphics Liabrary 和 MATHLAB C/C++ Math Liabrary 工具箱，就可以在 C/C++语言和 MATLAB 语言之间实现良好的交互，使其运算和图形处理功能更为强大。

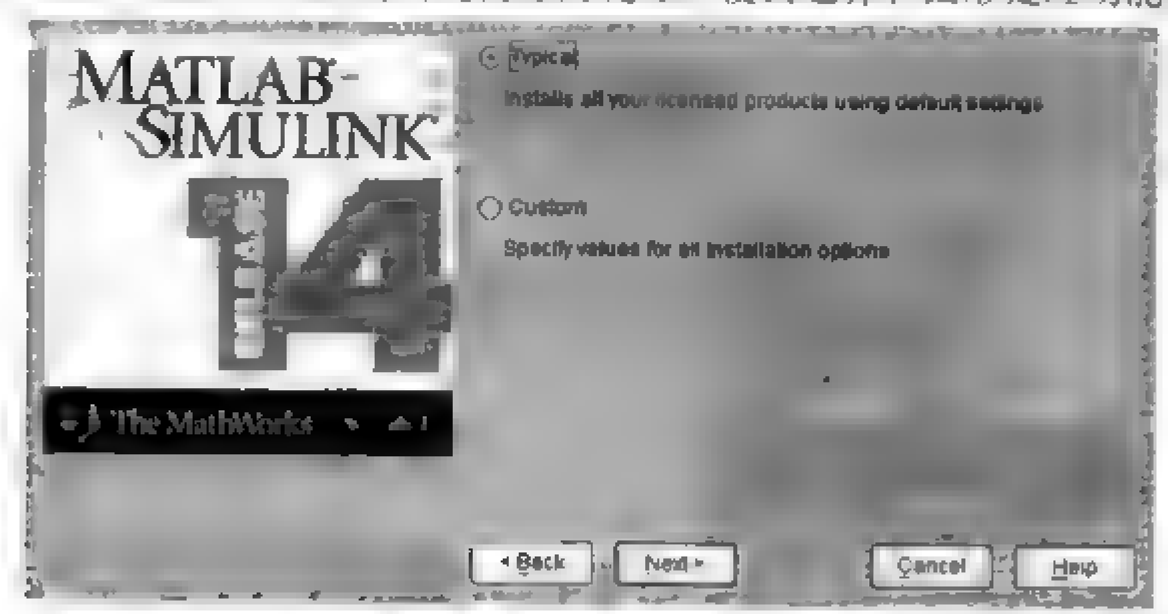


图 2-5　组件选择对话框

(6) 单击 Next 按钮，进入 MATLAB 7.0 的安装路径对话框，默认的路径为 C:\MATLAB7\，用户可以选择默认的路径，也可以单击 Browse 按钮选择新的安装路径。在此，选择为默认路径，如图 2-6 所示。

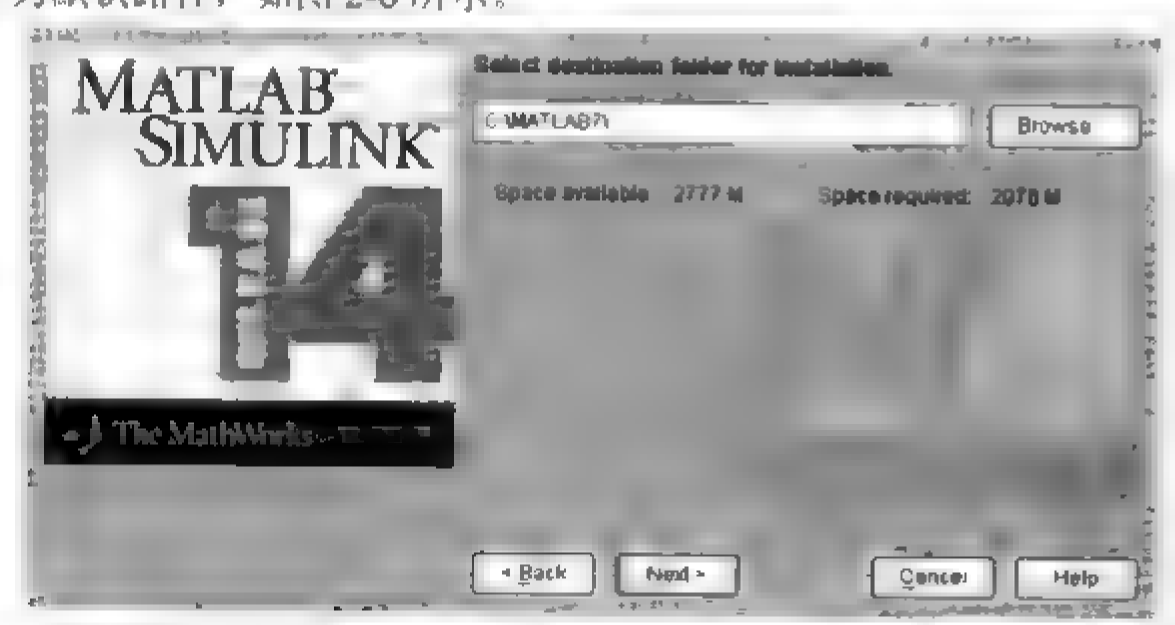


图 2-6　安装路径对话框

(7) 单击 MATLAB 7.0 安装路径对话框的 Next 按钮，进入 MATLAB 7.0 的安装状态，如图 2-7 所示。

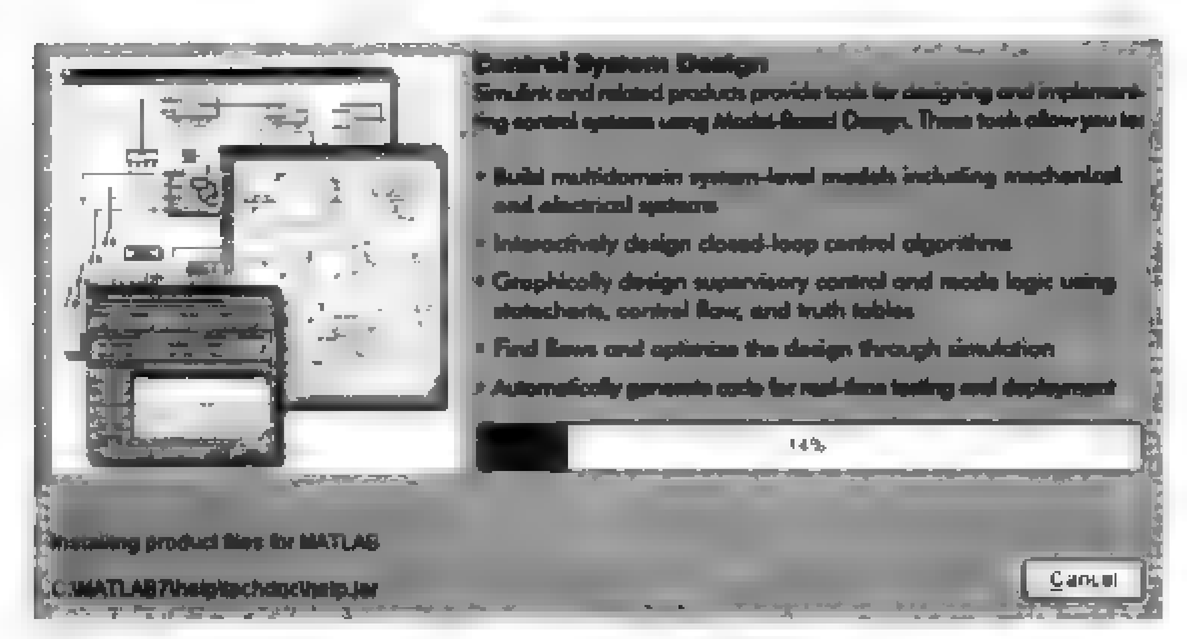


图 2-7　MATLAB 7.0 的安装状态

当安装到 100%时，系统会自动弹出 ProductConfiguration Notes 对话框，如图 2-8 所示。

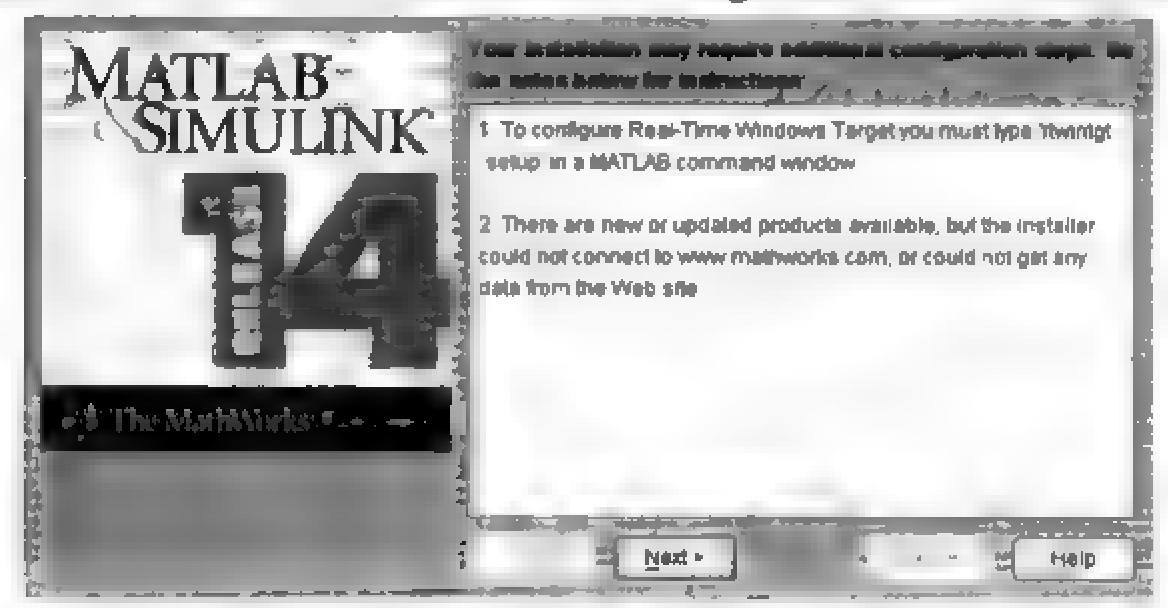


图 2-8　ProductConfiguration Notes 对话框

(8) 单击 ProductConfiguration Notes 对话框中的 Next 按钮，系统会弹出 Installation Complete 对话框。将单击 Finish 按钮，结束 MATLAB 7.0 的安装。选中 Start MATLAB 复选框，将在单击 Finish 按钮之后启动 MATLAB 7.0；否则，将不予启动，如图 2-9 所示。

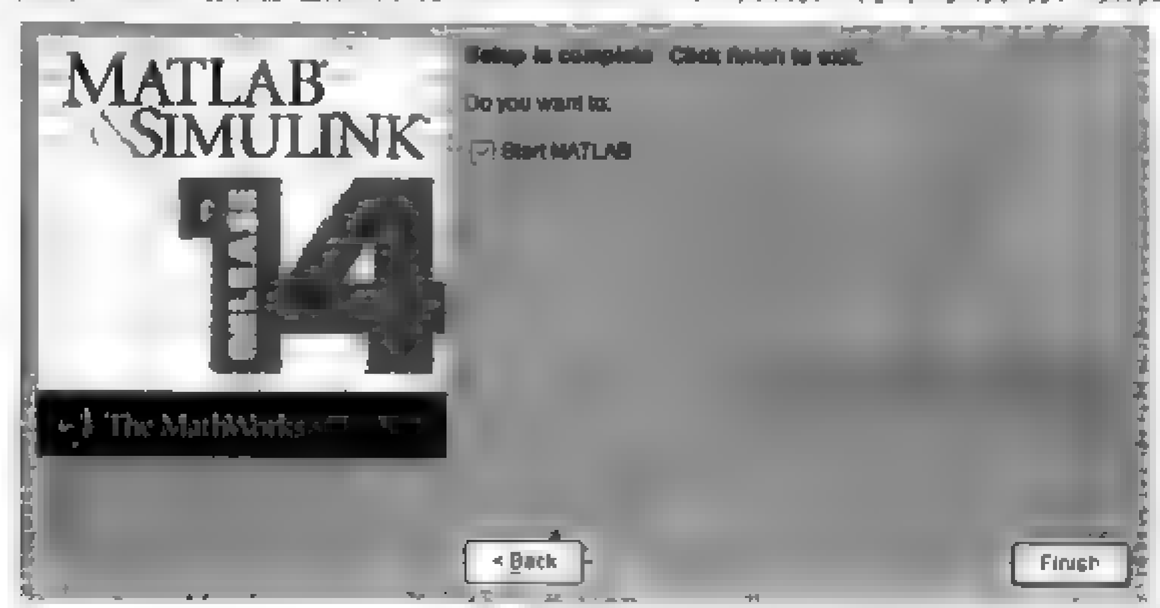


图 2-9　Installation Complete 对话框

至此，MATLAB 7.0 已经成功安装。

MATLAB 7.0 提供一个功能高度集成的可视化环境，其功能强大，操作方便。本章将主要介绍 MATLAB 7.0 基本桌面环境，以及如何实现 MATLAB 7.0 的路径搜索，并介绍如何使用 MATLAB 7.0 的帮助系统。这一章的内容是深入学习 MATLAB 7.0 的基础。

2.2　MATLAB 7.0 用户界面概述

在默认设置下，MATLAB 7.0 的用户界面通常包括有 6 个窗口，它们分别是 MATLAB 主窗口、命令窗口、命令历史窗口、当前目录窗口、发行说明书窗口和工作间管理窗口。对这些窗口的认识，是掌握 MATLAB 7.0 的基础，本节将主要介绍这些窗口基本知识。

2.2.1　启动 MATLAB 7.0

在正确完成安装并重新启动计算机之后，选择 Windows 桌面上的“开始”|“程序”|MATLAB 7.0|MATLAB 7.0 命令，或者直接双击系统桌面的 MATLAB 图标，启动 MATLAB 7.0，如图 2-10 所示。

图 2-10　启动 MATLAB 7.0

首次启动 MATLAB 7.0，将进入 MATLAB 7.0 默认用户界面，界面有 4 个主要的窗口：命令窗口、当前目录窗口、工作间管理窗口和发行说明书窗口。与以前的 MATLAB 版本不同的地方在于，MATLAB 7.0 除了有传统的主工具栏之外，还新增了快捷工具栏(Shortcut Toolbar)，如图 2-11 所示。

图 2-11　MATLAB 7.0 的用户界面

2.2.2　MATLAB 7.0 的主菜单

MATLAB 7.0 用户界面的主菜单如图 2-12 所示。关于主菜单中各个命令的详细应用，将在以后的章节具体介绍。

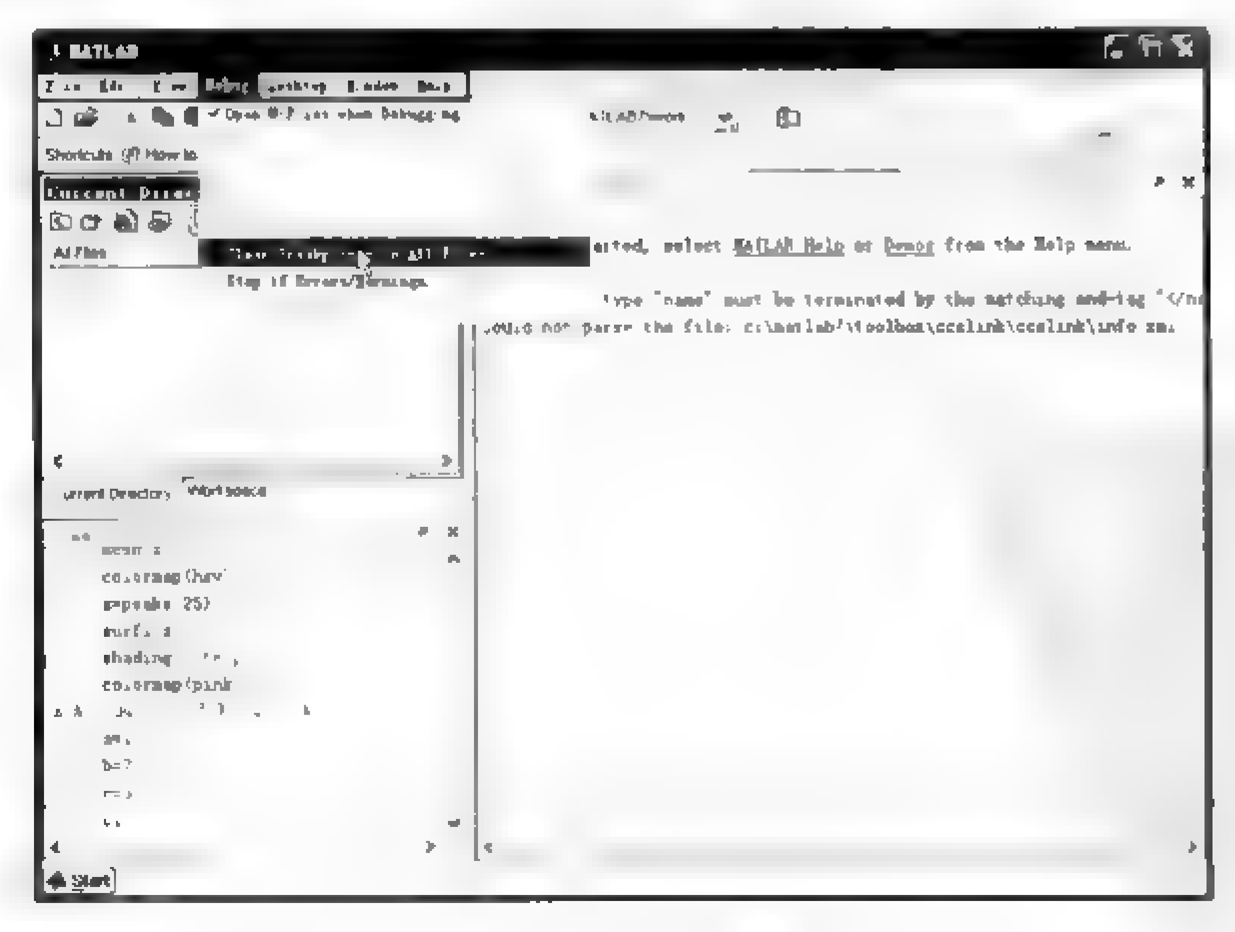


图 2-12　MATLAB 主菜单

2.2.3 MATLAB 7.0 的工具栏

MATLAB 7.0 用户界面的工具栏如图 2-13 所示。包括主工具栏和快捷工具栏，主工具栏中各个图标的功能如图 2-13 所示，关于快捷工具栏的详细应用，将在以后的章节具体介绍。

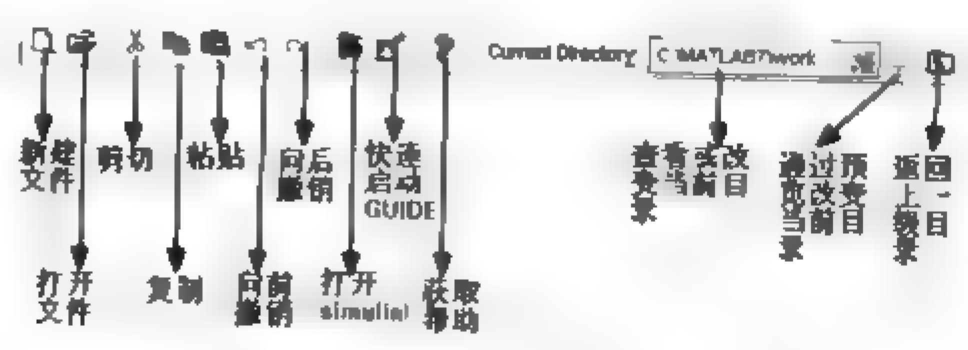


图 2-13 MATLAB 主工具栏

2.2.4 MATLAB 7.0 的窗口

打开 MATLAB 7.0，默认打开的窗口包括：(1)命令窗口(Command Window)；(2)命令历史窗口(Command History)；(3)工作间管理窗口(Workspace)；(4)当前路径窗口(Current Directory)。本节将介绍这些窗口的使用方法。此外，还有编译窗口、图形窗口和帮助窗口等其他种类的窗口，这些将在以后的章节中予以介绍。

1. 命令窗口(Command Window)

在默认设置下，命令窗口自动显示于 MATLAB 界面中，如果用户只想调出命令窗口，也可以选择 Desktop | Desktop Layout | Command Window Only 命令。MATLAB 7.0 用户界面的右侧窗口就为命令窗口。

命令窗口是和 MATLAB 编译器连接的主要窗口。“>>”为运算提示符，表示 MATLAB 处于准备状态，早期版本的 MATLAB 提示符为“？”。MATLAB 具有良好的交互性，当在提示符后输入一段正确的运算式时，只需按 Enter 键，命令窗口中就会直接显示运算结果。例如，计算一个圆的体积，假设球的半径为 10，那么只需在命令窗口中输入如下数据：

```
>> area=pi*10^2
```

按 Enter 键确认输入如图 2-14 所示，即可以得出结果如下：

```
area =
   314.1593
>>
```

同时 MATLAB 的提示符“>>”不会消失，这表明 MATLAB 继续处于准备状态。

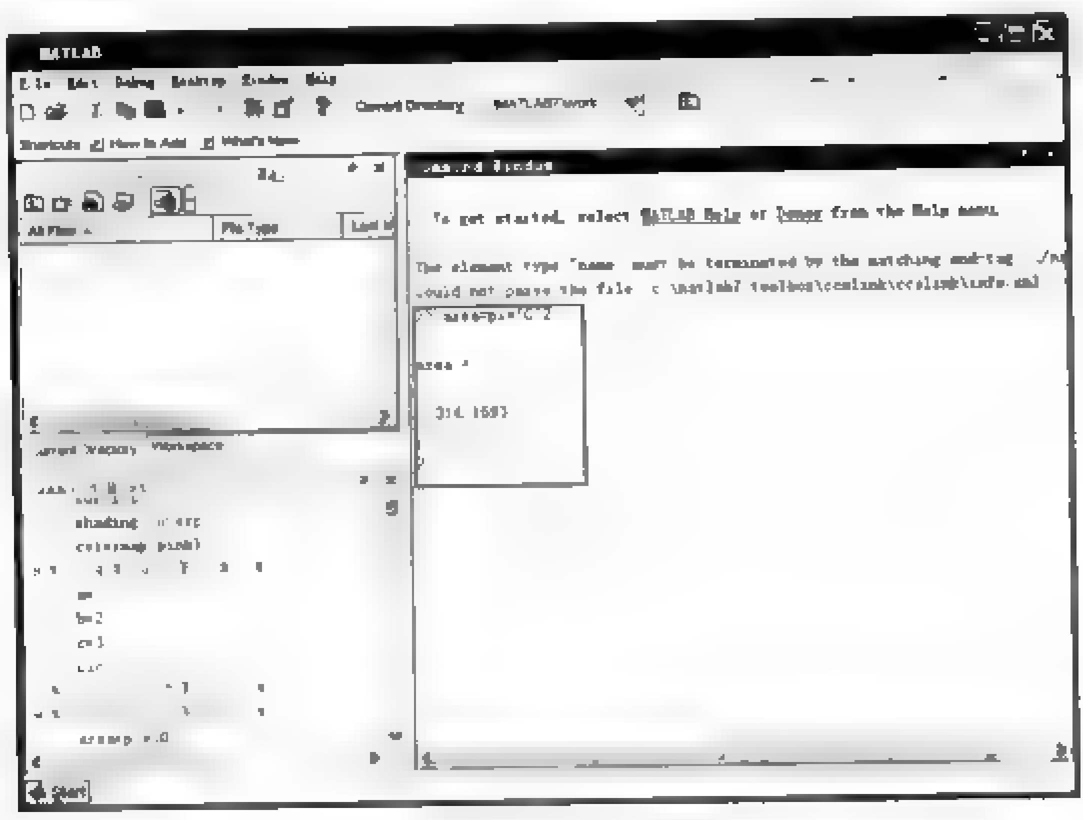


图 2-14 计算球的体积

2. 命令历史窗口(Command History)

在默认设置下，命令历史窗口自动显示于 MATLAB 界面中，用户也可以选择 Desktop|Command History 命令调出或隐藏该命令窗口。

命令历史窗口显示用户在命令窗口中所输入的每条命令的历史记录，并标明使用时间，这样可以方便用户的查询。如果用户想再次执行某条已经执行过的命令，只需在命令历史窗口中双击该命令；如果用户需要从命令历史窗口中删除一条或多条命令，只需选中这些命令，并单击右键，在弹出的快捷菜单中选择 Delete Selection 命令即可。命令历史窗口在早期的 MATLAB 版本中曾经有过雏形，并从 MATLAB 6.0 开始被赋予了更为强大的功能，其窗口形式如图 2-15 所示。

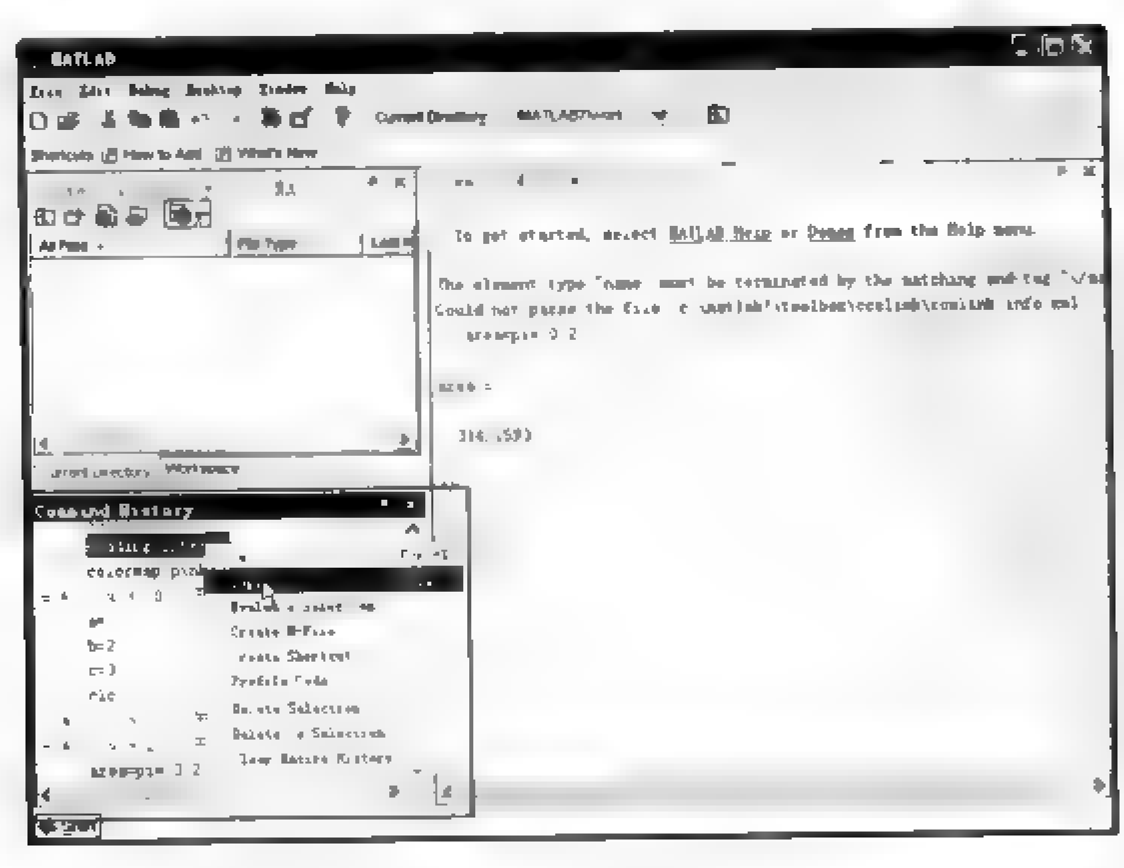


图 2-15 命令历史窗口

3. 工作间管理窗口(Workspace)

在默认设置下，工作间管理窗口自动显示于 MATLAB 界面中，用户也可以选择 Desktop| Workspace 命令调出或隐藏该命令窗口。

工作间管理窗口是 MATLAB 的重要组成部分，例如表达式 x=100 产生了一个名为 x 的变量，而且这个变量 x 被赋予 100 的值，这个值就被存储在计算机的内存中。工作间管理窗口就是用来显示当前计算机内存中 MATLAB 变量的名称、数学结构、该变量的字节数及其类型，在 MATLAB 中不同的变量类型对应不同的变量名图标，如图 2-16 所示。

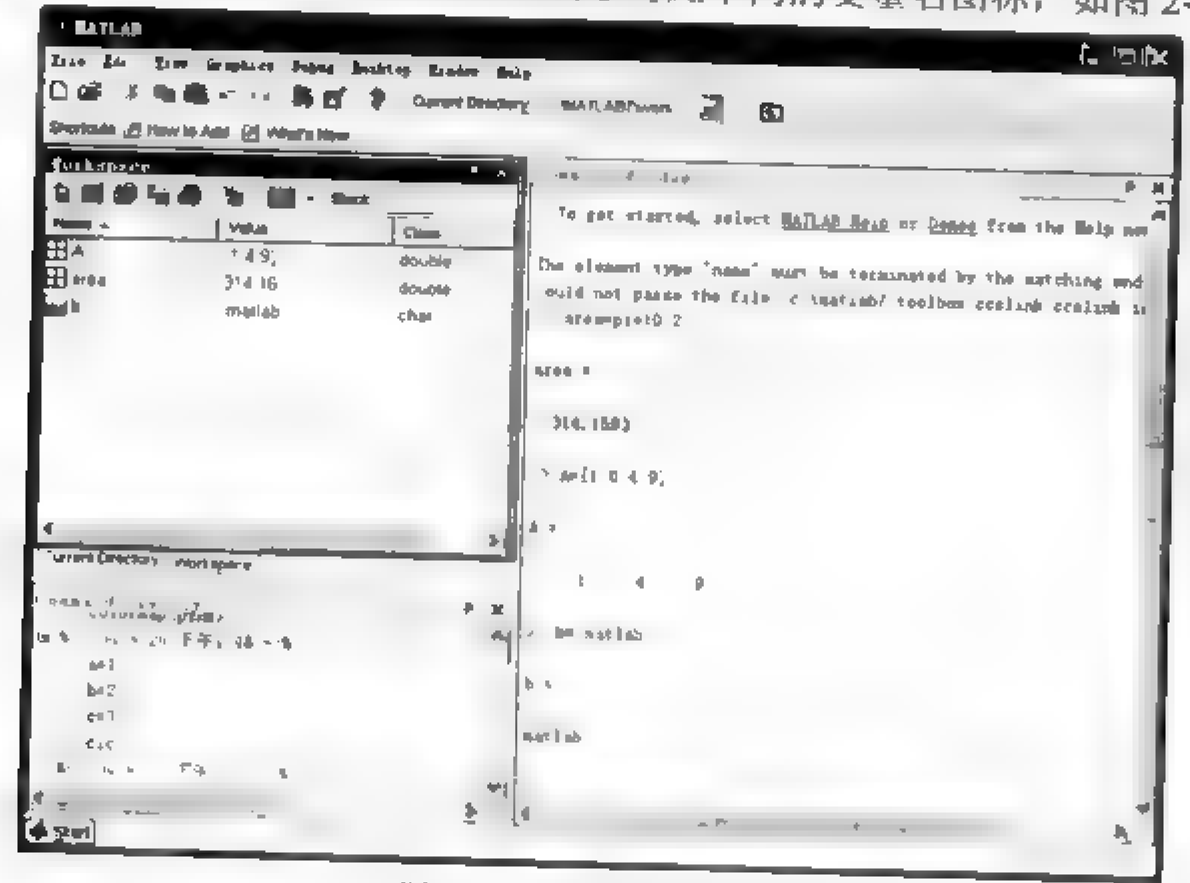


图 2-16　工作间管理窗口

注释：

在 MATLAB 命令窗口中运行的所有命令都共享一个相同的工作间，所以它们共享所有的变量，初学者应当重视。

4. 当前路径窗口(Current Directory)

在默认设置下，当前路径窗口自动显示于 MATLAB 界面中，用户也可以选择 Desktop| Current Directory 命令调出或隐藏该命令窗口。

当前路径窗口显示着当前用户工作所在的路径，以后本书还会单独对其应用做详细的介绍，窗口形式如图 2-17 所示。

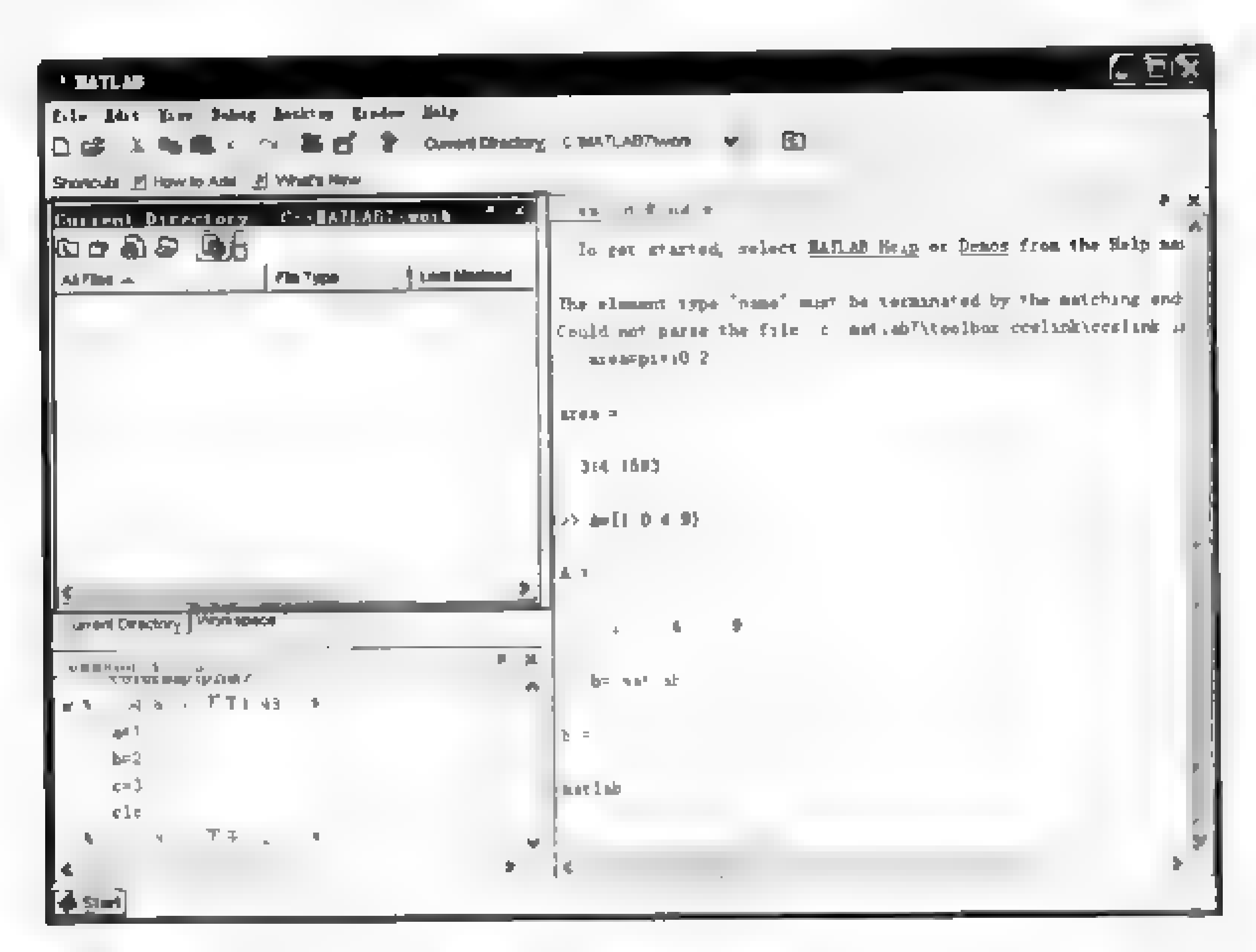


图 2-17　当前路径窗口

2.3　MATLAB 7.0 的路径搜索

MATLAB 7.0 有一个专门用于寻找“.m”文件的路径搜索器。“.m”文件是以目录和文件夹的方式分布于文件系统中的，一部分“.m”文件的目录是 MATLAB 7.0 的子目录，由于 MATLAB 7.0 的一切操作都是在它的搜索路径(包括当前路径)中进行的，所以如果调用的函数在搜索路径之外，MATLAB 7.0 就会认为此函数并不存在。

2.3.1　MATLAB 7.0 的当前目录

在命令窗口中输入 cd 命令，并按 Enter 键确认，即显示有当前 MATLAB 7.0 工作所在目录。

```
>> cd
D:\MATLAB\work
>>
```

2.3.2　MATLAB 7.0 的路径搜索

选择 MATLAB 的主窗口中 File Set Path 命令，如图 2-18 所示。

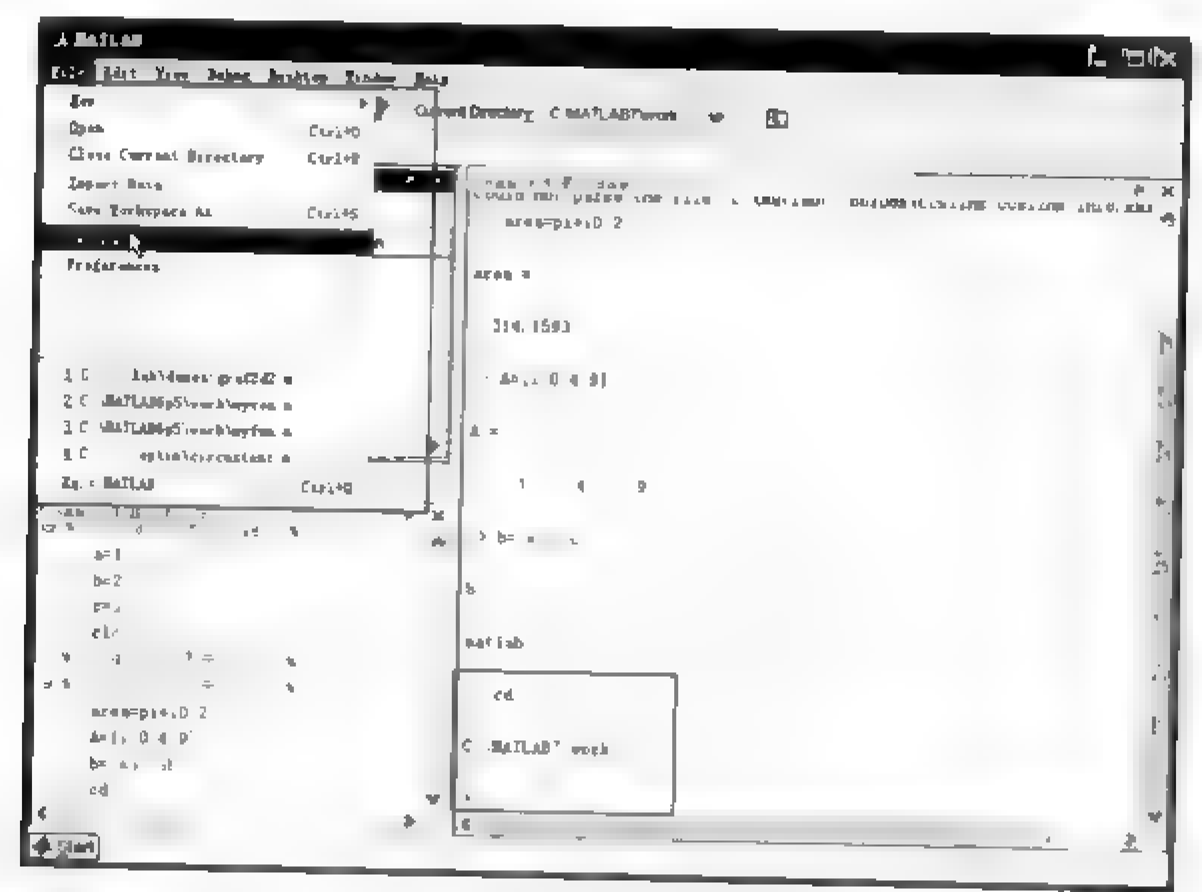


图 2-18　选择 Set Path 命令

由此进入到设置路径搜索的对话框，如图 2-19 所示。在对话框右侧的列表框中，列出的目录就是 MATLAB 7.0 的所有搜索路径。

如果只想把某一目录下的文件包含在搜索范围内忽略其子目录，则单击 Set Path 对话框中的 Add Folder 按钮，否则单击 Add with Subfolders 按钮，一般情况下选择后者。例如单击 Add Folder 按钮，此时系统会弹出如图 2-20 所示的“浏览文件夹”对话框，在这里我们选择 toolbox，最后单击“确定”按钮，则新目录将会出现在搜索路径中，新的搜索路径设置完毕。

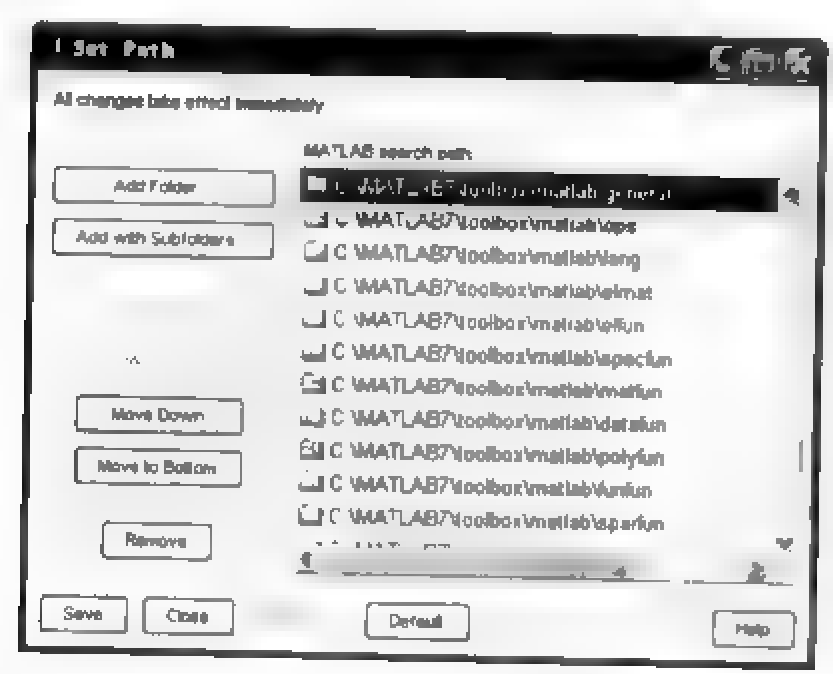


图 2-19　Set Path 对话框

图 2-20　“浏览文件夹”对话框

下面介绍几个路径搜索时常用的命令。

◆ path 命令

在命令窗口中输入 path 命令可以得到 MATLAB 7.0 所有的搜索路径，如下所示。

```
>> path

        MATLABPATH

    D:\MATLAB\work
    D:\MATLAB\toolbox\matlab\general
    D:\MATLAB\toolbox\matlab\ops
    D:\MATLAB\toolbox\matlab\lang
    D:\MATLAB\toolbox\matlab\elmat
    D:\MATLAB\toolbox\matlab\elfun
    D:\MATLAB\toolbox\matlab\specfun
    D:\MATLAB\toolbox\matlab\matfun
    D:\MATLAB\toolbox\matlab\datafun
    D:\MATLAB\toolbox\matlab\audio
……
```

◆ genpath 命令

在命令窗口中输入 genpath 命令可以得到 MATLAB 所有的搜索路径连接而成的一个长字符串，如下所示。

```
>> path

smod\import\standalone;D:\MATLAB\toolbox\physmod\import\standalone\solidworks;……
```

◆ editpath 或 pathtool 命令

在命令窗口中输入这两个命令中的任意一个就可以进入如图 2-21 所示的搜索路径设置对话框。

2.4　MATLAB 7.0 帮助系统的使用

有效地使用帮助系统所提供的信息，是用户掌握好 MATLAB 应用最佳途径。熟练的程序开发人员总会充分地利用软件所提供的帮助信息，而 MATLAB 的一个突出优点就是其拥有较为完善的帮助系统。MATLAB 的帮助系统可以分为联机帮助系统和命令窗口查询帮助系统。本节将对这两种帮助系统分别进行简要的介绍。

2.4.1　帮助窗口

MATLAB 7.0 的帮助窗口非常全面，几乎包括该软件的所有内容。选择 MATLAB 主窗口中 Help | MATLAB Help 命令(实际上选择 Help 下拉菜单中的前 4 项命令中的任何一项，均可进入帮助窗口)，如图 2-21 所示；也可以选择主窗口中的“？”按钮进入帮助窗

口；还可以在命令窗口中直接执行 helpwin、helpdesk 或 doc 命令进入帮助窗口。

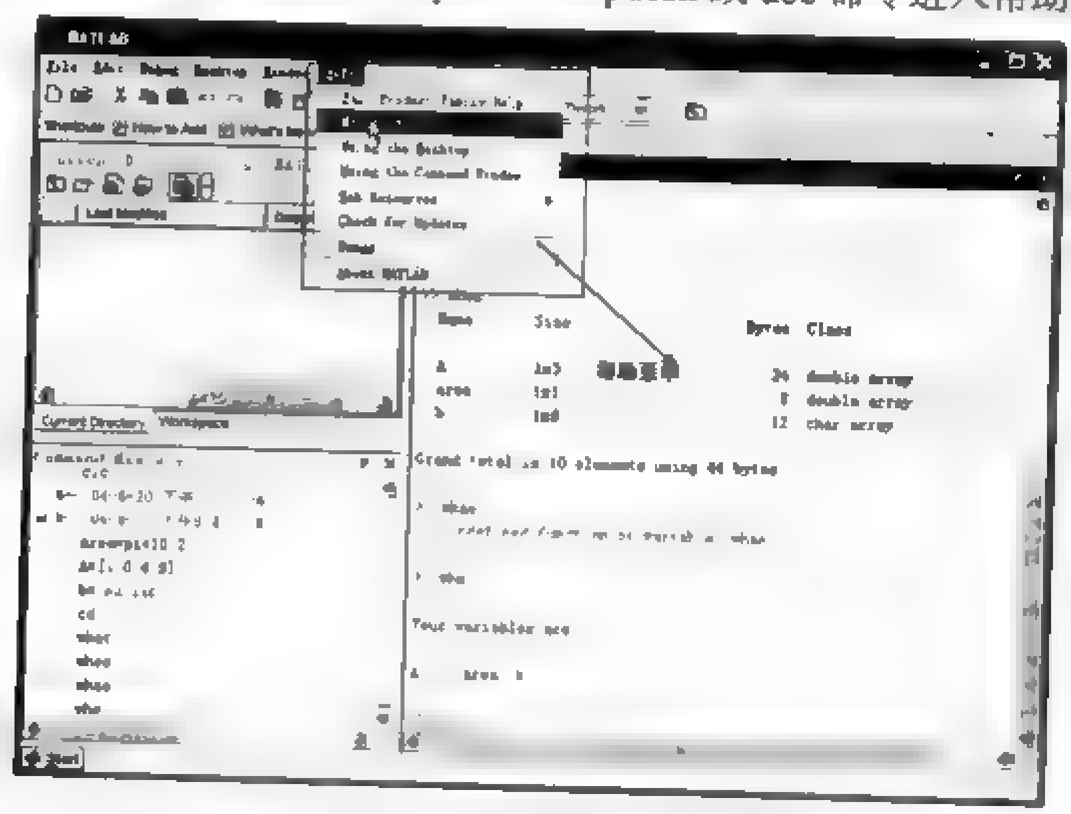


图 2-21　进入联机帮助系统

系统自动弹出帮助窗口，覆盖原有的默认窗口设置，如图 2-22 所示。关闭帮助窗口，系统又将回复原有的设置情况。

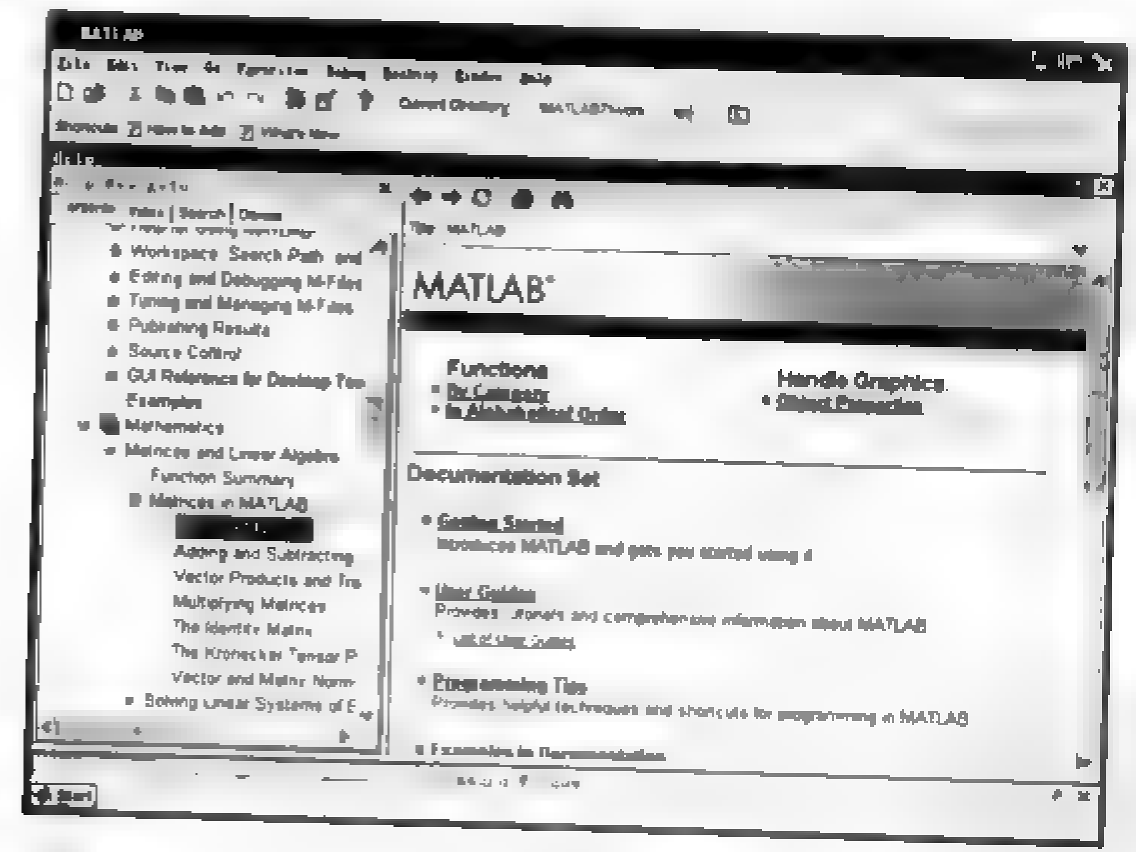


图 2-22　联机帮助系统

在联机帮助系统中，左侧部分为帮助导向界面，右侧为帮助显示界面。

帮助导向界面下侧的 4 个标签分别为帮助主题(Contents)、帮助索引(Index)、帮助查询(Search)和联机演示(Demos)。下面分别介绍其使用方法。

1. 帮助主题(Contents)

单击该按钮，将显示 MATLAB 的帮助内容，如图 2-22 所示。

2. 帮助索引(Index)

单击该按钮，出现如图 2-23 所示的对话框，在 Search index for 文本框中输入用户需要查找的内容，右边的窗口中会显示关于该内容的相关信息，如输入 plot 命令后，右侧窗口中立刻显示出相关的信息，如语法、描述和与其相关的函数等。

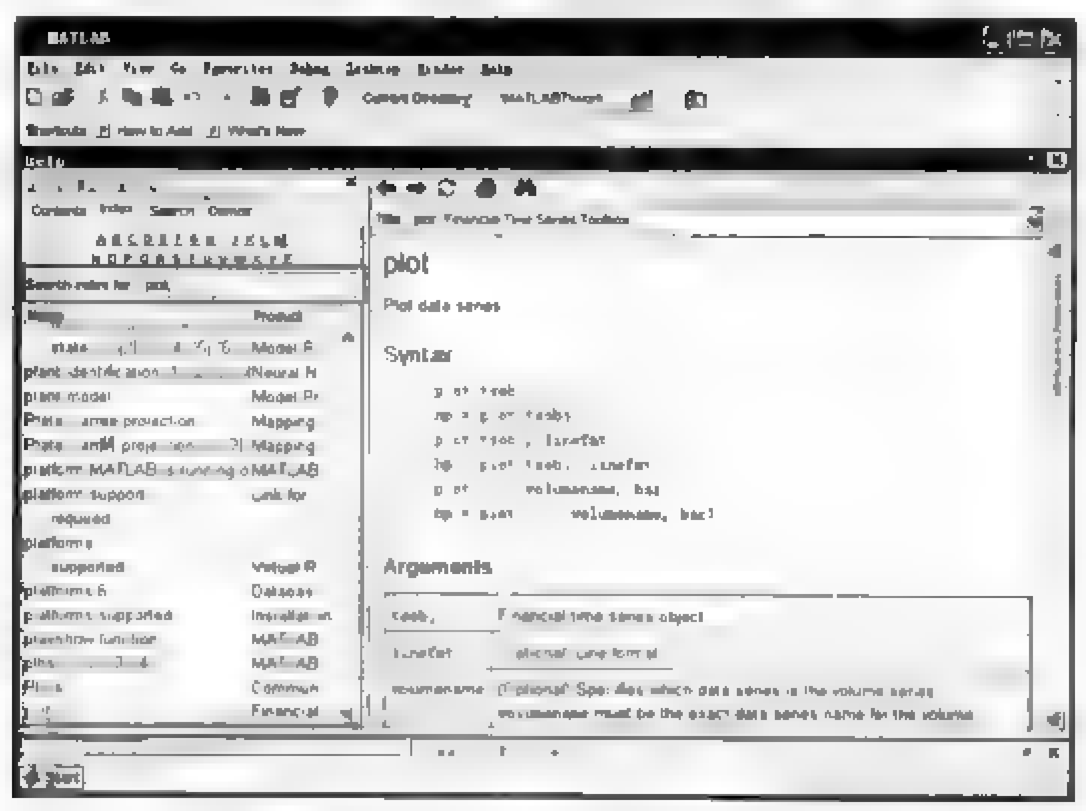


图 2-23　Index 功能的实现

3. 帮助查询(Search)

单击 Search 按钮，在 Search for 下拉列表框中选择或输入文件名，如在本例中输入函数名 plot，单击 Go 按钮，就会在右侧窗口中输出关于文件 plot 的相关信息，这些信息比上例中使用 Index 所查询的信息要丰富一些，如包括 plot 函数具体的实例和使用方法，如图 2-24 所示。

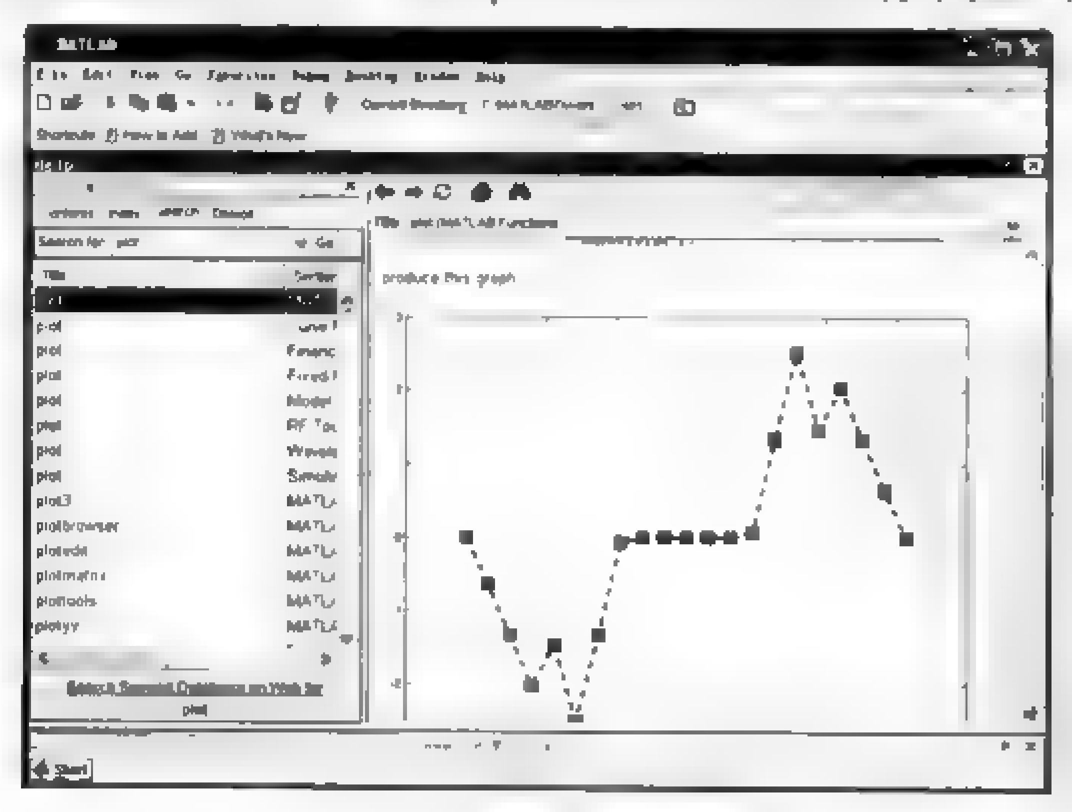


图 2-24　Search 功能的使用

4. 联机演示(Demos)

MATLAB 除了常规的帮助系统外，还设立了联机演示系统，对于初学者来说，查看 MATLAB 的联机演示是最佳的学习方法。在该项内容中，MATLAB 设置了许多关于各个工具箱内容的现成的程序，用户可以通过选择自己所需的部分来学习相关内容。

用户可以单击 Demos 按钮或者在命令窗口中运行 intro 进入 MATLAB 的联机演示界面，用户还可以通过选择 Help | Demos 命令进入联机演示界面。

联机演示窗口包含有两个部分，左边的部分是项目栏，用户可以用鼠标来选择所需演示的项目，右边是对此项目的说明文字。双击左边项目栏的具体内容，或者单击该项内容，再单击右边说明框中的 Run this demo 按钮，MATLAB 都将会弹出新的窗口进行联机演示操作。

如图 2-25 所示的就是在左边项目栏中选中 MATLAB | Graphics | 3-D Plots 后的结果。

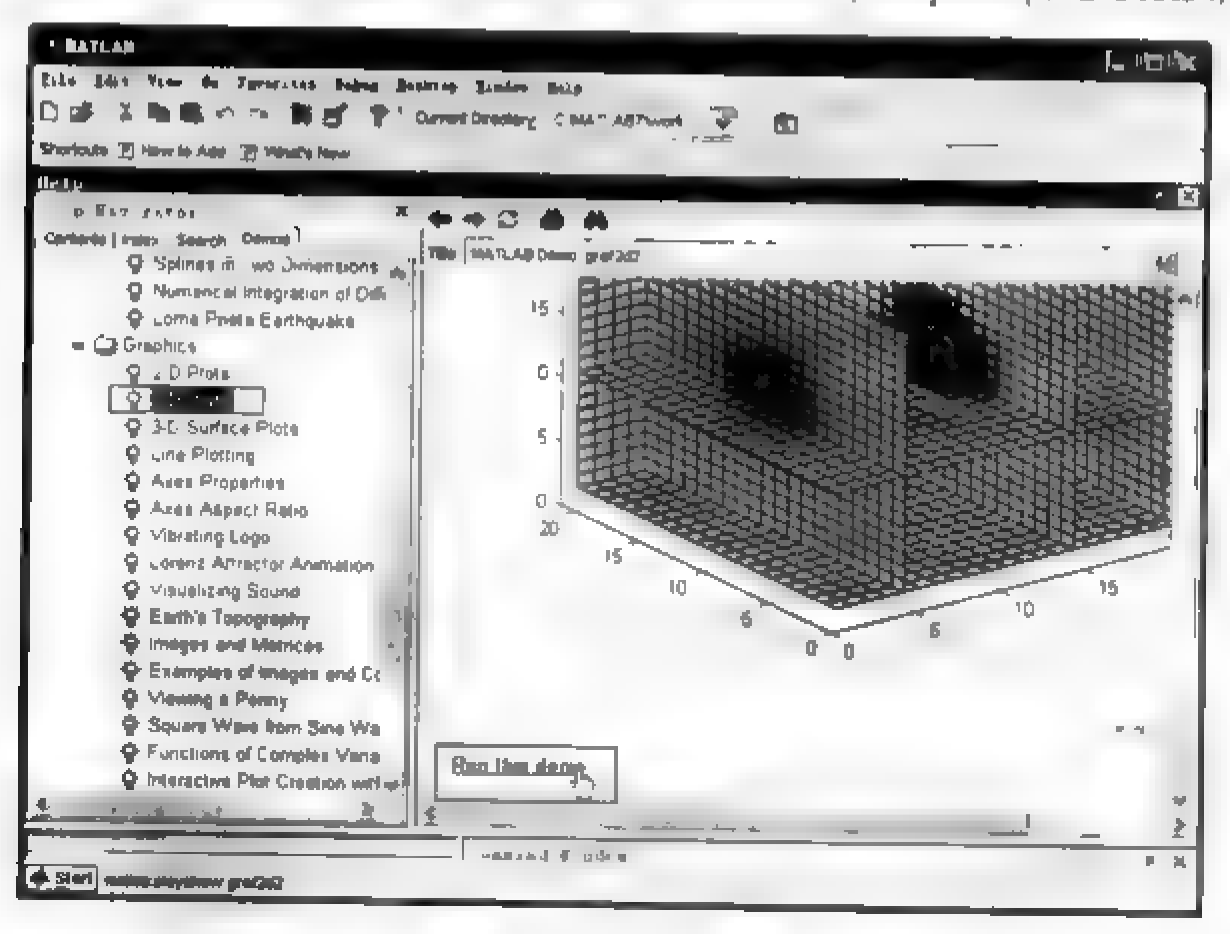


图 2-25　3-D plots 演示

双击图 2-25 中左侧的 3-D plots 按钮或者单击右侧的 Run this demo 按钮，弹出 Examples of X-Y Plots in MATLAB 对话框，如图 2-26 所示，该对话框主要分为 3 个部分，主体部分用于显示运算或者图形结果，下面部分用于显示对主体窗口结果的说明，右侧部分是控制按钮。本例用于介绍二维图形的绘制方法，因此，其主题窗口显示的是一个绘图框，下面部分的说明窗口显示的是该实例的使用方法。

```
% XYZ plots in MATLAB.
% Here are some examples of surface plots in MATLAB
```

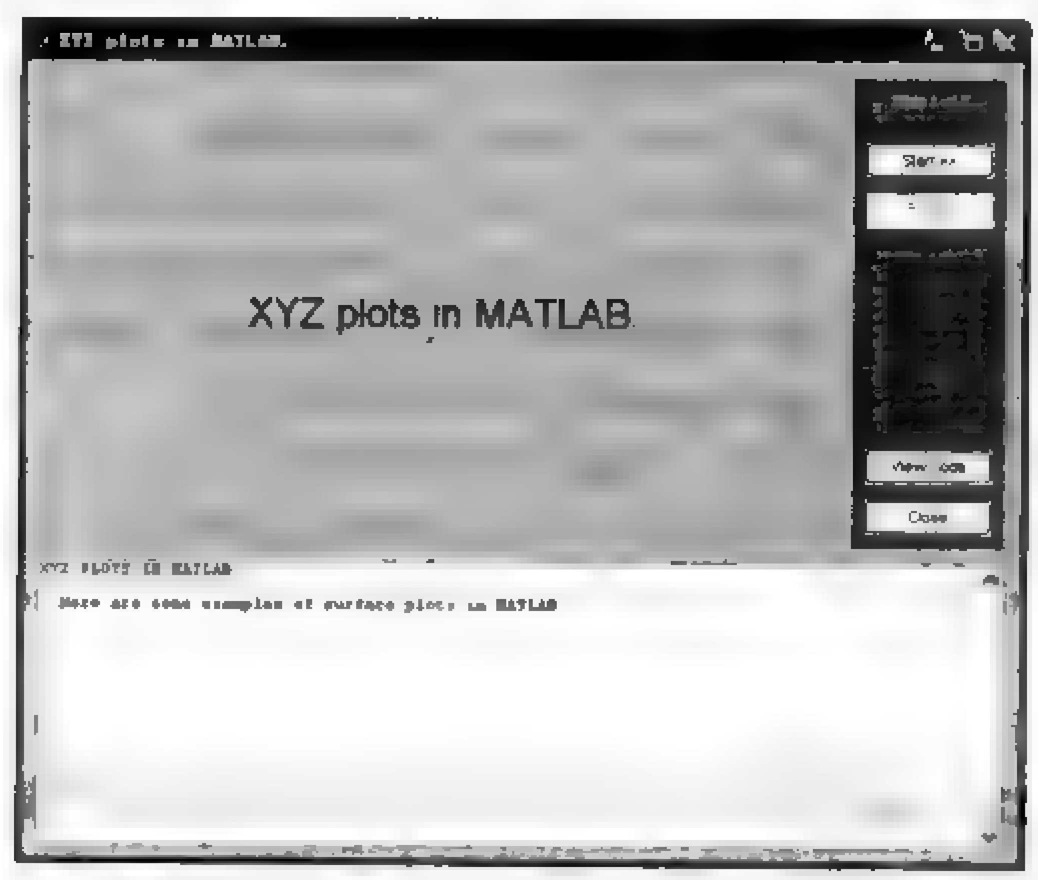


图 2-26　Examples of XZY Plots in MATLAB 对话框

单击 Examples of X-Y Plots in MATLAB 对话框的 Start 按钮，会出现如图 2-27 所示的运行结果，此时 MATLAB 的说明程序如下。

```
%% Mesh Plot of Peaks
z=peaks(25);
mesh(z);
colormap(hsv)
```

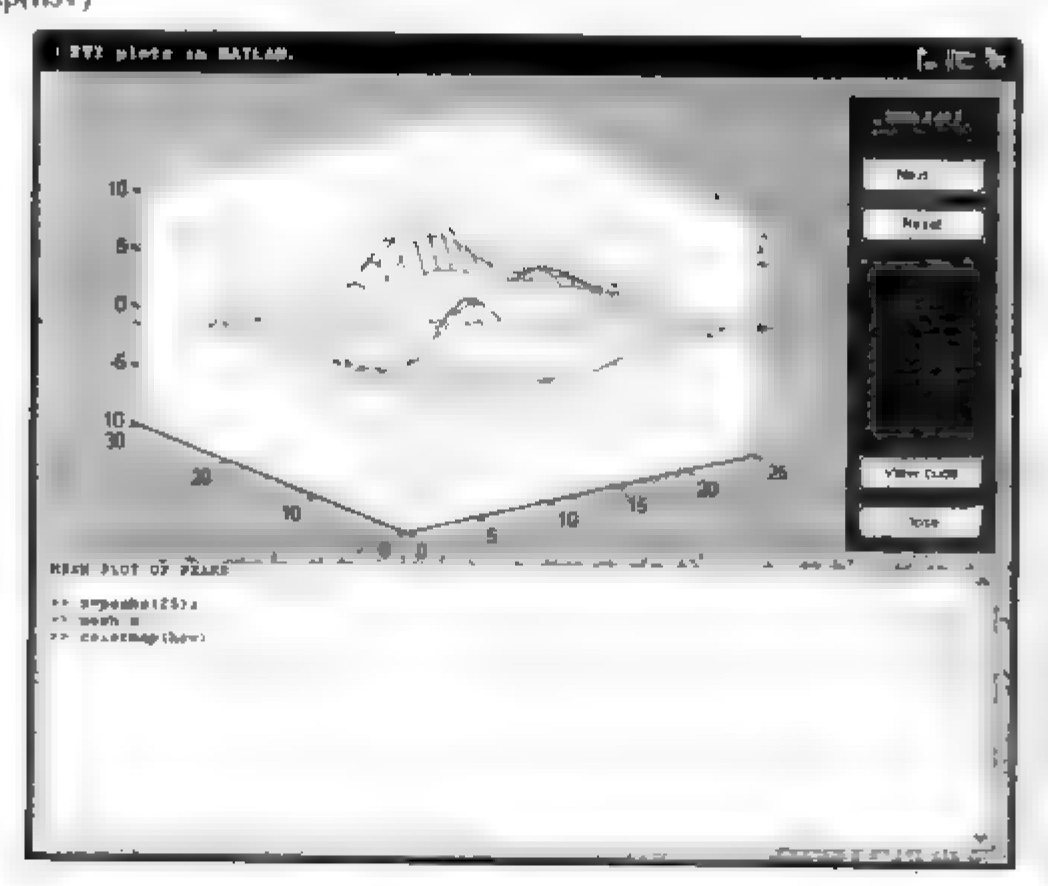


图 2-27　Demos 运行结果显示

继续单击 Start 按钮，可以演示该 Demos 的其他内容。单击 Reset 按钮，将返回该 Demos 的初始状态，用户可以进行其他的操作。单击 View code 按钮，将弹出一个编辑窗口，窗

口中有该 Domos 的源代码，如图 2-28 所示。

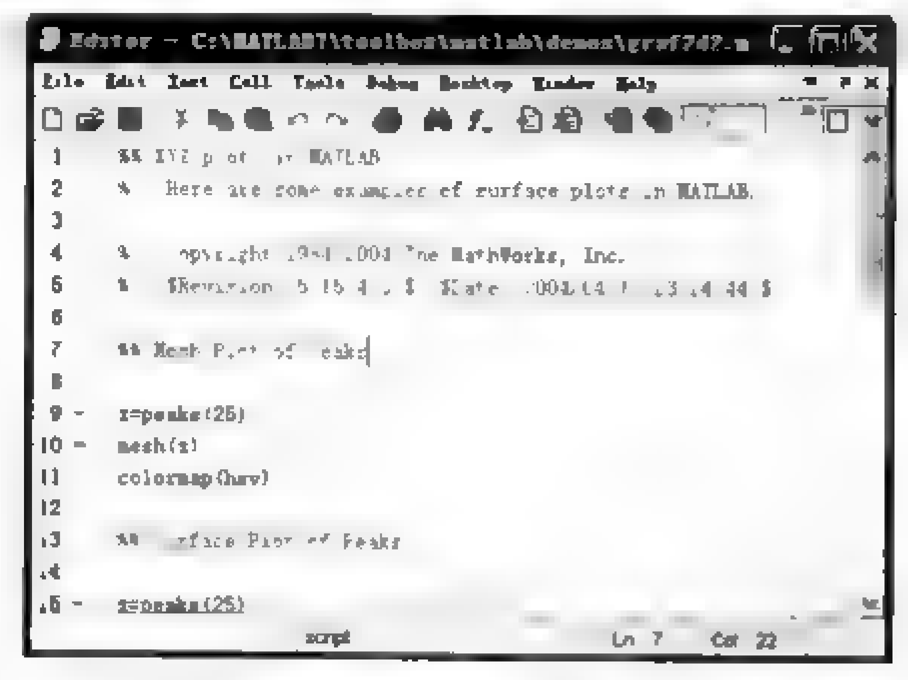


图 2-28　单击 View code 按钮调出的编辑窗口

2.4.2　命令窗口查询帮助

当用户对 MATLAB 有了一定了解之后，可以通过在命令窗口中直接输入命令来获得相关的帮助信息，这种获取方式比联机帮助更为快捷。在命令窗口中获取帮助信息的主要命令为 help 和 lookfor 函数。

1. help 函数

help 函数有 4 种用法，分别是 help、help+函数名(函数类名)、helpdesk 和 helpwin。其中对于后两者的使用本章已经做了叙述，在此只介绍前两者的应用。

◆ help 命令

在命令窗口中直接输入 help 命令，会显示当前的帮助系统中所包含的所有项目。需要注意的是用户在输入该命令后，命令窗口只显示当前搜索路径中的所有目录名称。

◆ help+函数名(函数类名)

当用户知道某个函数名称，如果想了解该函数的具体用法，只需在命令窗口中输入 help+函数名。例如用户想了解 det 函数(求行列式的值)的具体用法，只需在命令窗口中输入 help det，即可得到如下的关于此函数的基本信息。

```
>> help det
 DET     Determinant.
    DET(X) is the determinant of the square matrix X.
    Use COND instead of DET to test for matrix singularity.
    See also COND.
 Overloaded methods
    help gf/det.m
    help sym/det.m
>>
```

同样，当用户想知道某一函数类型的具体用法，只需在命令窗口中输入 help+函数类名即可。例如用户想得到 det 函数类型的具体用法，只需在命令窗口中输入 help det，即可得到如下的关于此函数类型的基本信息。

```
>>  help matfun
   Matrix functions - numerical linear algebra.
   Matrix analysis.
     norm        - Matrix or vector norm.
     normest     - Estimate the matrix 2-norm.
     rank        - Matrix rank.
     det         - Determinant.
     trace       - Sum of diagonal elements.
     null        - Null space.
     orth        - Orthogonalization.
     rref        - Reduced row echelon form.
     subspace    - Angle between two subspaces.
   Linear equations.
…………
     cholupdate  - rank 1 update to Cholesky factorization.
     qrupdate    - rank 1 update to QR factorization.
>>
```

2. lookfor 函数

一般来说，当用户知道某个函数的具体名称时，可以使用 help 函数寻找到相关的帮助信息，但是对于初学者来说，往往不知道函数的确切名称，在这种情况下 help 函数就无能为力，而使用 lookfor 函数可以很方便地解决这个问题。在使用 lookfor 函数时，用户只需知道某个函数的部分关键字，在命令窗口中输入 lookfor+关键字，就可以很方便地实现查找。例如，用户需要查找含有关键字 norm 的相关内容，即可以按照下述的方法得以实现。

```
>>  lookfor norm
RANDN    Normally distributed random numbers.
CONDEST 1-norm condition number estimate
NORM     Matrix or vector norm.
NORMEST Estimate the matrix 2-norm
NORMEST1 Estimate of 1-norm of matrix by block 1-norm power method.
LSQR     LSQR Implementation of Conjugate Gradients on the Normal Equations.
SPRANDN Sparse normally distributed random matrix.
…………
isnormempty.m: %ISNORMEMPTY
i_eval.m: % cgNormFunction\i_eval
mtableview.m: %CGNORMFUNCTION/MTABLEVIEW Displays the normfunction data.
return_data.m  %CGNORMFUNCTION/RETURN_DATA Returns model data for the points
specified by the inputs to this table
```

settingscheck.m: % CGNORMFUNCTION/SETTINGSCHECK : Perform a brief diagnostic to see whether we can evaluate this normaliser or not

……

2.5 习　　题

1. 在安装 MATLAB 7.0 软件时，试着有选择地安装各种组件。

2. 认识和理解 MATLAB 7.0 的各个窗口，使用 Desktop 菜单栏设置 MATLAB 7.0 的窗口布局。

3. 将 MATLAB 7.0 的当前路径设置到 C 盘根目录下。

4. 使用帮助窗口查找 plot 函数的帮助信息。

5. 使用 help 函数查找 plot3 函数的帮助信息。

6. 使用 lookfor 函数查找关于 gui 的帮助信息。

第3章 基本使用方法

如前所述，MATLAB 7.0 的优点不仅在于强大的功能，还在于其简单易学，本章主要是介绍 MATLAB 7.0 的基本使用方法，用户在学习完本章的内容后，可以进行基本的数值运算，从而能够容易地解决许多在学习和科研中遇到的计算问题。

3.1 简单的数学运算

1. 数学式的输入

MATLAB 7.0 的命令窗口给用户提供了一个很好的交互式平台，当命令窗口处于激活状态时，会出现提示符“>>”，在提示符的右边有一个闪烁的光标，这表示 MATLAB 7.0 正处于准备状态，等待用户输入各种命令。

MATLAB 7.0 最主要的功能是数值计算，对于简单的数值计算来说，使用 MATLAB 7.0 可以很轻松地解决。下面首先介绍在 MATLAB 7.0 中的基本数值运算符号，如表 3-1 所示。

表 3-1 数值运算符号

符号	功能	实例
+	加法	1+2
-	减法	1-2
*	乘法	1*2
/、\	除法	1/2 或是 2\1
^	乘方	2^1

下面介绍几种基本数值计算的方法。

◆ 直接输入法

在命令窗口中直接输入数学表达式，按 Enter 键确认，即可得到结果。

例 3-1 光明小学一年级有 4 个班，每班 30 人，二年级有 3 个班，每班 35 人，求该小学一二年级一共有多少人。

解：在 MATLAB 中直接输入计算表达式即可，如下所示。

```
>> 4*30+3*35
ans =
    225
>>
```

该例生成的结果被 MATLAB 7.0 自动赋予变量名 ans，它是 answer 的简写。

◆ 存储变量法

采用直接输入法虽然简单易行，但是当用户需要解决的问题较复杂时，采用直接输入法有时将变得比较困难，此时，可以通过采用给变量赋予变量名的方法来进行操作。

例 3-2 使用存储变量法再求例 3-1。

解：可以将一年级的总人数命名为 grade1，二年级的总人数命名为 grade2，两个年级的总人数命名为 total。在 MATLAB 7.0 命令窗口中输入如下命令，并按 Enter 键确认。

```
>> grade1=4*30
grade1 =
    120
>> grade2=3*35
grade2 =
    105
>> total=grade1+grade2
total =
    225
>>
```

在本例中，先分别求出一二年级的总人数，并将结果分别赋予变量 grade1 和 grade2。再将变量 grade1 和 grade2 相加，得到两个年级的总人数。这就避免了总是使用同一个变量 ans 的问题。

由此可见，采用以上两种方法所得结果是一样的，使用第一种方法比较直接和简单，而使用第二种方法思路更清晰。

在大多数情况下，MATLAB 7.0 语言对空格不予处理。此外，在 MATLAB 7.0 的表达式中，遵守四则运算法则，即乘法和除法要优先于加减法；而指数运算等又优先于乘除法，而括号的运算级别更高；在有多层括号存在的情况下，从括号的最里边向最外边逐渐扩展。在 MATLAB 中，小括号代表着运算级别，而中括号则一般用于生成矩阵。

2. 标点符号的使用

在 MATLAB 7.0 语言中，标点符号的使用相对比较灵活，不同的标点符号代表不同的运算，或是被赋予了特定的含义。如表 3-2 中就列出了一些在 MATLAB 7.0 中常用的标点符号。

表 3-2 MATLAB 7.0 中常用的标点

标点符号	定　义	标点符号	定　义
；	区分行，取消运行显示等	.	小数点以及域访问等
，	区分列，函数参数分隔符等	...	连接语句
:	在数组中应用较多	'	字符串的标识符号
()	指定运算优先级等	=	赋值符号
[]	矩阵定义的标志等	!	调用操作系统运算
{}	用于构成单元数组等	%	注释语句的标识

下面，对其中几个比较常用的符号进行简单介绍。

◆ 分号(;)

如上节所示，在命令窗口输入命令后，如直接按 Enter 键，将在命令窗口直接显示这条命令的计算结果。有时，用户要求禁止显示计算的中间结果，在 MATLAB 7.0 中，可以使用分号来实现此项功能。

例 3-3　使用分号重新计算例 3-2。

```
>> grade1=4*30;
>>   grade2=3*35;
>>   total=grade1+grade2
total =
    225
>>
```

对比例 3-2 和例 3-3，可以看出使用分号之后，计算的中间结果将不再显示，但是计算的结果相同。

◆ 百分号(%)

有时，为了增强程序的可读性，需要给一些语句添加注释语句，在 MATLAB 7.0 中，使用百分号(%)来进行句子的注释操作，百分号之后的所有文本都将看作是注释。

例 3-4　给例 3-3 的程序段增添注释。

```
>> grade1=4*30              %求一年级的总人数
grade1 =
    120
>> grade2=3*35              %求二年级的总人数
grade2 =
    105
>> total=grade1+grade2   %一二年级的人数和
total =
    225
>>
```

上例中，使用百分号对一些程序段进行了注释，可以看出，使用注释语句之后，没有对计算结果产生任何影响，但是，却极大地增强了程序的可读性，这在编写大型程序或是多人合作编写程序时显得尤为重要。

◆ 逗号(,)

MATLAB 7.0 允许用户在一行中输入多个命令语句，这些语句使用逗号或是分号隔开，它们的区别在于使用逗号时，命令语句的运行结果将予以显示，而使用分号时，运行结果将予以隐藏。

例 3-5　逗号和分号的综合运用。

```
>> x=sin(1),y=cos(1);z=tan(1),w=atan(1)
x =
     0.8415
z =
     1.5574
w =
     0.7854
>>
```

上边的程序中，第一行输入了4条语句，同时使用了逗号和分号，当命令语句后边使用逗号或是不使用标点符号时，命令的执行结果将在命令窗口中予以显示，如本例中的x、z和w；而当使用分号时命令的执行结果将在命令窗口中予以隐藏，如本例中的y。

♦ 续行号(…)

在编写程序时，往往会碰到命令行很长的情况，此时，为了使程序看起来清晰或是阅读起来方便，可以将程序分成多行分别书写，在MATLAB 7.0中，使用3个连续的句号(…)来实现此项功能。

例 3-6　续行号的使用。

```
>> grade1=4*...
30            %求一年级的总人数
grade1 =
    120
```

如上边程序所示，当3个句号出现在数学运算符号和变量之间时，就起到了连接语句的作用。但是，并不是将续行号放到任何地方都可以起到连接作用的，在以下这些情况下，使用续行号将起不到预定效果。

例 3-7　续行号的错误使用方式。

```
>> grade2=...3*35;            %求二年级的总人数
grade2=...
??? grade2=...
             |
Error: The expression to the left of the equals sign is not a valid target for an assignment.
>>
```

在上边的程序中，由于续行号直接放在等号之后，所以MATLAB 7.0提示表达式非法。

```
>> total=grade1+gra...
de2
??? de2
Error: Missing MATLAB 7.0 operator.
>>
```

在上段程序中，续行号置于变量名grade2中间，结果MATLAB 7.0将其不作处理，

因此不能起到续行号的作用，当再输入 det2 时，MATLAB 7.0 将提示没有输入运算符号。

3　常用的操作命令和键盘技巧

在使用 MATLAB 7.0 语言编制程序时，掌握一些常用的操作命令和键盘操作技巧，可以起到事半功倍的效果,分别如表 3-3 和表 3-4 所示。

表 3-3　常用的操作命令

命　令	该命令的功能	命　令	该命令的功能
cd	显示或改变工作目录	hold	图形保持命令
clc	清除工作窗	load	加载指定文件的变量
clear	清除内存变量	pack	整理内存碎片
clf	清除图形窗口	path	显示搜索目录
diary	日志文件命令	quit	退出 MATLAB 7.0
dir	显示当前目录下文件	save	保存内存变量到指定文件
disp	显示变量或文字内容	type	显示文件内容
echo	工作窗信息显示开关		

表 3-4　常用的键盘操作和快捷键

键盘按钮和快捷键	该操作的功能	键盘按钮和快捷键	该操作的功能
↑(Ctrl+p)	调用上一行	Home(Ctrl+a)	光标置于当前行开头
↓(Ctrl+n)	调用下一行	End(Ctrl+e)	光标置于当前行结尾
←(Ctrl+b)	光标左移一个字符	Esc(Ctrl+u)	清除当前输入行
→(Ctrl+f)	光标右移一个字符	Del(Ctrl+d)	删除光标处字符
Ctrl+←	光标左移一个单词	Backspace(Ctrl+h)	删除光标前字符
Ctrl+→	光标右移一个单词	Alt+BackSpace	恢复上一次删除

3.2　MATLAB 7.0 的数据类型

MATLAB 7.0 的数据类型包括数字、字符串、矩阵、单元型和结构型变量，本节将重点介绍这些常用的数据类型。

3.2.1　常量和变量

1. 常量

在 MATLAB 中有一些特定的变量，它们已经被预定义了某个特定的值，因此这些变

量被称为常量。MATLAB 7.0 中的常量主要有 pi、inf 和 eps 等，如表 3-5 所示。

表 3-5　MATLAB 7.0 的常用常量

常　量	常量的功能	常　量	常量的功能
ans	用作结果的默认变量名	nargin	函数的输入参数个数
beep	使计算机发出“嘟嘟”声	nargin	函数的输出参数个数
pi	圆周率	varagin	可变的函数输入参数个数
eps	浮点数相对误差	varagout	可变的函数输出参数个数
inf	无穷大	realmin	最小的正浮点数
NaN 或 nan	不定数	realmax	最大的正浮点数
i 或 j	复数单位	bitmax	最大的正整数

下面，对其中几个常量的用法进行简单介绍。

◆ inf

在 MATLAB 7.0 语言中，Inf 表示无穷大。MATLAB 7.0 允许的最大数是 2^{1024}，超过该数时，系统将会视该数为无穷大，其他的软件在出现数据无穷大时可能会出现死机的情形，而 MATLAB 7.0 则会给出用户警告信息。同时用 inf 代替无穷大，但不会导致死机的情形，这也就是 MATLAB 7.0 的一个重要优点。比如用户在命令窗口中输入 1/0，结果如下。

```
>> 1/0
Warning. Divide by zero.
(Type "warning off MATLAB 7.0:divideByZero" to suppress this warning.)
ans =
    inf
>>
```

◆ eps

eps 用来判断是否为 0 元素的误差限。一般情况下，用到的误差限 MATLAB 7.0 函数默认为 eps，它的值大约为 2.2204e-16。

◆ pi

pi 用来表示圆周率的数值。在命令窗口中输入 pi，可以得到如下结果。

```
>> pi
ans =
     3 1416
>>
```

◆ 纯虚数

常用的纯虚数用 i 或 j 来表示，也就是数学上的 sqrt(-1)。如果在程序中没有专门给这

两个变量定义，那么系统将默认它们为单位虚数，用户可以直接使用；如果用户在程序中对它们有了新的定义，则这两个变量将保留新值。比如，在命令窗口中输入 i，按 Enter 键确认，结果如下。

```
>> i
ans =
          0 + 1.0000i
>>
```

如果用户先定义 i=1，那么得到的结果如下。

```
>> i=1;
>> i
i =
     1
>>
```

当然用户也可以将其他任意变量设定为 sqrt(-1)。例如可以在命令窗口中输入 xushu=sqrt(-1)，则变量 xushu 将被赋值为 sqrt(-1)。

2. 变量

变量是 MATLAB 7.0 的基本元素之一，与其他常规程序设计语言不同的是 MATLAB 7.0 语言不要求对所使用的变量进行事先说明，而且它也不需要指定变量的类型，系统会根据该变量被赋予的值或对该变量所进行的操作来自动确定变量的类型。

在 MATLAB 7.0 语言中，变量的命名有如下规则。

- 变量名长度不超过 31 位，超过 31 位的字符系统将忽略不计
- 变量名区分大小写
- 变量名必须以字母开头，变量名中可以包含字母、数字或下划线，但不允许出现标点符号

需要注意的是，用户如果在对某个变量赋值时，该变量已经存在，系统则会自动使用新值代替旧值。例如在命令窗口中输入“a=1;a=2”命令，则得到如下结果。

```
>> a=1;
>> a=2
a =
     2
>>
```

此外，MATLAB 7.0 中提供了丰富的运算函数，用户只需正确调用其形式就可以得到满意的结果，常见的运算函数如表 3-6 所示。

表 3-6　MATLAB 7.0 常用函数表

函 数 名	函 数 功 能	函 数 名	函 数 功 能
sin	正弦	pow2	以 2 为底幂函数
sinh	双曲正弦	sqrt	平方根
asin	反正弦	nextpow2	不小于变量的最小 2 指数
asinh	反双曲正弦	abs	模
cos	余弦	tanh	双曲正切
cosh	双曲余弦	atan	反正切
acos	反余弦	atan2	四象限反正切
acosh	反双曲余弦	atanh	反双曲正切
tan	正切	sec	正割
acoth	反双曲余切	sech	双曲正割
exp	指数	asec	反正割
log	自然对数	asech	反双曲正割
log10	以 10 为底对数	csc	余割
log2	以 2 为底对数	csch	双曲余割
acsc	反余割	besselh	第三类 bessel 函数
acsch	反双曲余割	besseli	改进的第一类 bessel 函数
cot	余切	besselk	改进的第二类 bessel 函数
coth	双曲余切	beta	β
acot	反余切	betainc	不完全的 β
angle	相角	betaln	β 的对数
conj	复共轭	ellipj	Jacobi 椭圆函数
imag	复矩阵虚部	cross	向量叉积
real	复矩阵实部	ellipke	完全椭圆积分
unwrap	打开相角	erf	误差函数
isreal	实阵判断	erfc	补充的误差
cplxpair	调整数为共轭对	erfcx	比例补充的误差
fix	向零方向舍入	erfinv	反误差函数
floor	向负方向舍入	expint	幂积分
ceil	向正方向舍入	gamma	Gamma 函数
round	四舍五入	gammainc	不完全 Gamma 函数
mod	有符号求余	gammaln	Gamma 函数的对数
rem	无符号求余	legendre	联合 legendre 函数
sign	符号函数	besselj	第一类 bessel 函数
airy	Airy 函数	bessely	第二类 bessel 函数

3.2.2　浮点数和复数

1. 浮点数

几乎在所有的情况下，MATLAB 7.0 的数据都是以双精度数值来表示的，这些双精度数在系统内部用二进制来表示。这是计算机通常的表示数据的方式，但也带来了一些问题，比如有很多实数不能被精确地表示，对能够表示的值也有一个限制，并且还存在一个浮点相对误差限。所谓相对误差限是指 MATLAB 7.0 语言能够区分两个不同大小的数时，这两个数之间的最小差值。

例 3-8　浮点数的精度。

```
>> a=0.33-0.5+0.17
a =
  2.7756e-017
>> b=0.33+0.17-0.5
b =
     0
>> c=0.17-0.5+0.33
c =
  5.5511e-017
>>
```

在上例中，虽然人工可以很轻易地算出这 3 个式子的计算结果是相同的，但是由于这些数字都是使用二进制存储的，在使用双精度数来表达这些数时，往往就会出现一些误差，而且这些误差通常无法避免。但是，这些误差会小于误差限 eps，在 MATLAB 7.0 中，eps 值为 2.2204e-016，用户可以在命令窗口中直接输入 eps 值就可以得到结果。

2. 复数

MATLAB 7.0 语言对复数的处理也是十分简便的，在处理复数问题时，不需要进行其他任何的附加操作。

例 3-9　复数的表示方法。

```
>> a1=pi+3.14i
a1=
   3.1416 + 3.1400i
>> a2=pi+3.14j
a2 =
   3.1416 + 3.1400i
>>
>> b=4*(1+3/sqrt(-1))
b =
   4.0000 -12.0000i
```

```
>> c=sqrt(-1)
c =
            0 + 1.0000i
>> d=sin(pi)i
??? d=sin(pi)i
Error: Missing MATLAB 7.0 operator.          %出现错误
>> d= >> sin(pi)
ans =
   1.2246e-016
>>
```

在上面的例子中，使用i、j和sqrt(-1)都生成了复数单位，需要注意的是，当直接使用sin(pi)i时，由于sin(pi)i没有任何意义，所以MATLAB 7.0提示缺少运算符，但是使用sin(pi)*i则可以得到正确结果。可见，只有数字才能与字符i和j直接相连，而表达式则不可以。

例3-10　复数的数学运算。

```
>> a=1+2*i
a =
    1.0000 + 2.0000i
>> b=3-4i
b =
    3.0000 - 4.0000i
>> c=pi+sin(pi/2)*i
c =
    3.1416 + 1.0000i
>> d=a+b
d =
    4.0000 - 2.0000i
>> e=a*d
e =
    8.0000 + 6.0000i
>> f=a/e
f =
    0.2000 + 0.1000i
>> g=a^f
g =
    1.0040 + 0.3127i
>>
```

3.3　习　　题

1. 计算 $\sin(3)+e^2$。

2. 设 $u=1$，$v=3$，计算以下习题。

(1) $4\frac{u^2}{3v}$

(2) $\frac{(u+\cos(v))^2}{v-u}$

(3) $\frac{\sqrt{u-3v}}{3v}$

(4) $\frac{\pi}{3}\cos(\pi)$

3. 仿照例 3-8，试验一下在对浮点数使用不同的运算顺序时，是否会对运算结果产生不同的影响。

4. 计算如下表达式的值。

(1) 11/5+6

(2) (11/5)+6

(3) 11/(5+6)

(4) 3^2^3

(5) 3^(2^3)

(6) (3^2)^3

(7) round(-11/5)+6

(8) ceil(-11/5)+6

(9) floor(-11/5)+6

5. 计算下列表达式的值。

(1) (3－5i)(4+3i)

(2) sin(1.2)(2-9i)

第4章　数值计算功能

本章将介绍MATLAB 7.0的数值计算功能，包括MATLAB 7.0的向量、矩阵和数组，并介绍它们之间的运算，此外，还介绍了一些特殊的矩阵数据结构。通过对本章的学习，读者可以编写简单且功能完善的MATLAB 7.0程序，从而解决各类基本问题，用户可以通过本章逐步掌握MATLAB 7.0的数值计算方法。

4.1　向量及其运算

向量是组成矩阵的基本元素之一，MATLAB 7.0提供了关于向量运算的强大功能。本节将对向量的基本知识和在MATLAB 7.0中的应用进行比较详细的介绍。

4.1.1　向量的生成

1. 在命令窗口中直接输入向量

在MATLAB 7.0中，生成向量最简单的方法就是在命令窗口中按一定格式直接输入。输入的格式要求是，向量元素用“[ ]”括起来，元素之间用空格、逗号或者分号相隔。需要注意的是，用它们相隔生成的向量形式是不相同的：用空格或逗号生成行向量；用分号生成列向量。

例4-1　在命令窗口中直接输入向量。

解：在命令窗口中输入如下命令，并按Enter键确认。

```
>> a1=[11 14 17 18];
>> a2=[11,14,17,18];
>> a3=[11;14;17;18];
>> a1
a1 =
    11    14    17    18
>> a2
a2 =
    11    14    17    18
>> a3
a3 =
    11
    14
```

```
        17
        18
>>
```

注释：

MATLAB 7.0 可以在行和列向量之间进行转置，使用的命令为“'”，如 a1'=a3，读者可以自己尝试转置操作。

2. 等差元素向量的生成

当向量的元素过多，同时向量各元素有等差的规律，此时采用直接输入法将过于繁琐。针对该种情况，MATLAB 7.0 提供了几种简易的输入方法，分别介绍如下。

- 冒号(:)生成法：基本格式为向量 $Vec = vec0:n:Vecn$，其中 Vec 表示生成的向量，$Vec0$ 表示第一个元素，n 表示步长，$Vecn$ 表示最后一个元素。
- 使用 linspace 函数：这是一个线性等分向量函数，基本格式为 $Vec = linespace(Vec0,Vecn,n)$，其中 Vec 表示生成的向量，$Vec0$ 表示第一个元素，$Vecn$ 表示最后一个元素，n 表示生成向量元素的个数。当 n 为默认时，系统将默认为 100。

例 4-2　等差元素向量的生成。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> vec1=10:5:50
vec1 =
    10    15    20    25    30    35    40    45    50
>> vec2=50:-5:10
vec2 =
    50    45    40    35    30    25    20    15    10
>> vec3=linspace(10,50,6)
vec3 =
    10    18    26    34    42    50
>>
```

4.1.2　向量的基本运算

向量的基本运算包括向量与数的四则运算、向量与向量之间的加减运算、向量之间的点积、向量之间的叉积和向量之间的混合积等。下面将对它们分别予以介绍。

1. 向量与数的四则运算

- 向量与数的加法(减法)：向量中的每个元素与数的加法(减法)运算。

例 4-3　向量与数的加法和减法。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> vec1=linspace(10,50,6)
vec1 =
     10     18     26     34     42     50
>> vec1+100
ans =
    110    118    126    134    142    150
>>
```

◆ 向量与数的乘法(除法)：向量中的每个元素与数的乘法(除法)运算。

例 4-4　向量与数的乘法和除法。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> vec2=logspace(0,10,6)   %对数等分向量
vec2 =
  1.0e+010 *
     0.0000      0.0000      0.0000      0.0001      0.0100      1.0000
```

当进行除法运算时，向量只能作为被除数，数只能作为除数。例如：

```
>> vec2/1000
ans =
  1.0e+007 *
     0.0000      0.0000      0.0000      0.0001      0.0100      1.0000
>>
```

2. 向量与向量之间的加减运算

向量与向量的加法(减法)运算：向量中的每个元素与另一个向量中相对应的元素的加法(减法)运算。

例 4-5　向量与向量的加法和减法。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> vec1=linspace(10,50,6)
vec1 =
     10     18     26     34     42     50
>> vec2=logspace(0,2,6)
vec2 =
     1.0000      2.5119      6.3096     15.8489     39.8107    100.0000
>> vec3=vec1+vec2
vec3 =
    11.0000     20.5119     32.3096     49.8489     81.8107    150.0000
>>
```

3. 点积、叉积和混合积

◆ 向量的点积：由数学知识可知，两个向量的点积等于其中一个向量的模与另一个向

量在这个向量的方向上的投影的乘积。在 MATLAB 7.0 中，提供有专门计算向量点积的函数 dot。需要注意的是，点积生成的是一个数，在 MATLAB 7.0 中计算向量的点积还要注意各向量维数的一致性，以保证点积的操作合法。

例 4-6　计算向量 $x1 = (11,22,33,44)$ 与向量 $x2 = (1,2,3,4)$ 的点积。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> x1=[11 22 33 44]
x1 =
    11    22    33    44
>> x2=[1,2,3,4]
x2 =
     1     2     3     4
>> a=dot(x1,x2)
a =
   330
>> sum(x1.*x2)                %还可以采用 sum 方法计算向量的点积
ans =
   330
>>
```

- 向量的叉积：由数学知识可知，叉积的几何意义是指过两个相交向量的交点，并与此两向量所在平面垂直的向量。在 MATLAB 7.0 中，同样提供有专门计算向量叉积的函数 cross。应当注意的是，在 MATLAB 7.0 中计算向量的点积也要注意各向量维数一致(由几何意义可知，向量维数只能为 3)，以保证叉积的操作合法。

例 4-7　计算相关向量的叉积。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> x1=[11 22 33 44]
x1 =
    11    22    33    44
>> x2=[1,2,3,4]
x2 =
     1     2     3     4
>> x3=cross(x1,x2)
??? Error using ==> cross
A and B must have at least one dimension of length 3.
>> x1=[11 22 33]
x1 =
    11    22    33
>> x2=[1 2 3]
x2 =
     1     2     3
>> x3=cross(x1,x2)
```

```
x3 =
     0     0     0
>>
```

从例 4-7 可以看出，当计算叉积的向量维数为 4 时，系统会提示出错。

- 向量的混合积：由数学知识可知，向量的混合积的几何意义是它的绝对值表示以向量为棱的平行六面体的体积。它的符号按照右手法则定义。在 MATLAB 7.0 中，向量的混合积由以上介绍的两个函数 dot 和 cross 得以实现。应当注意的是，在求向量的混合积时，函数的顺序不可颠倒，否则将会出现错误。

例 4-8　计算向量的混合积。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> a=[1 2 3]
a =
     1     2     3
>> b=[2 4 3]
b =
     2     4     3
>> c=[5 2 1]
c =
     5     2     1
>> v=dot(a,cross(b,c))
v =
   -24
>> v=cross(a,dot(b,c))
??? Error using ==> cross
A and B must be same size.
>>
```

从例 4-8 可以看出，在求向量的混合积时，函数的顺序不可颠倒，因为 dot 函数产生的是一个数，它不能用来和另外的向量进行叉积运算。所以在例 4-8 中的最后部分，当出现该种情况时，MATLAB 7.0 给出错误提示。

4.2　矩阵及其运算

MATLAB 语言是由早期专门用于矩阵运算的计算机语言发展而来的，其名称就是“矩阵实验室”(Matrix Laboratory)的缩写。MATLAB 语言最基本、最重要的功能就是进行实数矩阵或是复数矩阵的运算，其所有的数值功能都以矩阵为基本单元来实现。矩阵是 MATLAB 的最重要的部分，将对矩阵及其运算进行详细介绍。

4.2.1　矩阵的生成

矩阵的生成有多种方式，通常使用的有 4 种方法：

- 在命令窗口中直接输入矩阵
- 通过语句和函数产生矩阵
- 在 M 文件中建立矩阵
- 从外部的数据文件中导入矩阵

其中在命令窗口中直接输入矩阵是最简单、最常用的创建数值矩阵的方法。比较适合于创建较小的简单矩阵，把矩阵的元素直接排列到方括号中，每行内的元素用空格或逗号相隔，行与行之间的内容用分号相隔。

例 4-9　在命令窗口中直接输入矩阵。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> matrix=[1 ,1, 1, 1;2, 2, 2, 2;3, 3, 3, 3;4, 4, 4, 4]   %采用逗号形式相隔
matrix =
     1     1     1     1
     2     2     2     2
     3     3     3     3
     4     4     4     4
>> matrix=[1 1 1 1;2 2 2 2;3 3 3 3;4 4 4 4]            %采用空格形式相隔
matrix =
     1     1     1     1
     2     2     2     2
     3     3     3     3
     4     4     4     4
>>
```

4.2.2　矩阵的基本数值运算

矩阵的基本运算通常包含有矩阵与常数的四则运算、矩阵与矩阵之间的四则运算以及矩阵的逆运算等，本小节将对矩阵的这些运算形式做简要的介绍。

1. 矩阵与常数的四则运算

矩阵与常数的四则运算即是指矩阵各元素与常数之间的四则运算。在矩阵与常数进行除法运算时，常数通常只能做为除数。

例 4-10　练习矩阵与常数的四则运算。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> matrix=[1 1 1 1;2 2 2 2;3 3 3 3;4 4 4 4]
matrix =
```

```
      1      1      1      1
      2      2      2      2
      3      3      3      3
      4      4      4      4
>> m1=100+matrix
m1 =
    101    101    101    101
    102    102    102    102
    103    103    103    103
    104    104    104    104
>> m2=100-matrix
m2 =
     99     99     99     99
     98     98     98     98
     97     97     97     97
     96     96     96     96
>> m3=100*matrix
m3 =
    100    100    100    100
    200    200    200    200
    300    300    300    300
    400    400    400    400
>> m4=matrix/2
m4 =
     0.5000     0.5000     0.5000     0.5000
     1.0000     1.0000     1.0000     1.0000
     1.5000     1.5000     1.5000     1.5000
     2.0000     2.0000     2.0000     2.0000
>>
```

2. 矩阵之间的四则运算

◆ 矩阵与矩阵的加法(减法)

矩阵与矩阵的加法(减法)即是指矩阵各元素之间的加法(减法)运算。矩阵必须具有相同的阶数时才可以进行加法(减法)运算。

例 4-11　矩阵之间的加法和减法运算。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> matrix=[1 1 1 1;2 2 2 2;3 3 3 3;4 4 4 4]
matrix =
      1      1      1      1
      2      2      2      2
      3      3      3      3
      4      4      4      4
>> m1=20*matrix
```

```
m1 =
    20    20    20    20
    40    40    40    40
    60    60    60    60
    80    80    80    80
>> m2=m1+matrix
m2 =
    21    21    21    21
    42    42    42    42
    63    63    63    63
    84    84    84    84
>> m3=[11 22 33;1 2 3;4 5 6]
m3 =
    11    22    33
     1     2     3
     4     5     6
>> m4=matrix-m1
m4 =
   -19   -19   -19   -19
   -38   -38   -38   -38
   -57   -57   -57   -57
   -76   -76   -76   -76
>> m5=m3+m1
??? Error using ==> +
Matrix dimensions must agree.
>>
```

由例 4-11 可以看出，矩阵 *m3* 为 3×3 阶，而 *m1* 为 4×4 阶，因而如果求 *m5* = *m3* + *m1*，MATLAB 7.0 将提示出错误信息，表示不能进行 *m5* = *m3* + *m1* 运算。

◆ 矩阵与矩阵的乘法

在 MATLAB 7.0 中，矩阵的乘法使用运算符“*”。由数学知识，如果 A 是一个 $m\times s$ 阶矩阵，B 是一个 $s\times n$ 矩阵，那么规定矩阵 A 与矩阵 B 的乘积是一个 $m\times n$ 矩阵。必须注意的是，只有当第一个矩阵(左矩阵)的列数等于第二个矩阵(右矩阵)的行数时，两个矩阵的乘积才有意义。

例 4-12　矩阵的乘法运算。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> A=[1 1 1 1;2 2 2 2;3 3 3 3;4 4 4 4]
A =
     1     1     1     1
     2     2     2     2
     3     3     3     3
     4     4     4     4
```

```
>> B=[1 5 9 2;6 3 5 7;2 5 8 9;4 5 6 3]
B =
     1     5     9     2
     6     3     5     7
     2     5     8     9
     4     5     6     3
>> C=A*B
C =
    13    18    28    21
    26    36    56    42
    39    54    84    63
    52    72   112    84
>> D=[1 5 9 ;6 3 5 ;2 5 8 ]
D =
     1     5     9
     6     3     5
     2     5     8
>> E=A*D
??? Error using ==> *
Inner matrix dimensions must agree.
>>
```

由例 4-12 可以看出，矩阵 A 为 4×4 阶，而 D 为 3×3 阶，两者的阶数不符合乘法的要求，因此 MATLAB 7.0 将提示出错误信息，表示不能进行 $E = A*D$ 运算。

◆ 矩阵与矩阵的除法

在 MATLAB 7.0 中，矩阵的除法有左除和右除两种，分别以符号“\”和“/”表示。通常矩阵除法可以用来求解方程组的解，因此在使用 MATLAB 7.0 进行矩阵除法运算时，也必须保证各矩阵的数学合理性，否则 MATLAB 7.0 将无法进行运算。

一般情况下，$X = A\backslash B$ 表示 $A*X = B$ 的解，而 $X = A\backslash B$ 表示 $X*A = B$ 的解。从 MATLAB 6.0 开始，矩阵的左除和右除的区别在逐渐减少。

由线性代数知识可知，对于方程组 $AX = B$，其解的存在性需要满足一定的条件，关于这些内容在 MATLAB 7.0 中的使用，本书将在后面的章节再做详细介绍。

例 4-13　矩阵的除法运算：求解方程组 $A*X = B$ 的解。

解：在 MATLAB 7.0 命令窗口中进行直接操作。

```
%本例中 A=[2 1 -1;2 1 0;1 -1 1];B= B=[1 -1 3;4 3 2]。
>> A=[2 1 -1;2 1 0;1 -1 1]
A =
     2     1    -1
     2     1     0
     1    -1     1
>> B=[1 -1 3;4 3 2]
B =
```

```
     1    -1     3
     4     3     2
>> X=B/A
X =
   -2.0000    2.0000    1.0000
   -2.6667    5.0000   -0.6667
```

4.2.3 矩阵的特征参数运算

在进行科学计算的运算时，要对矩阵进行大量的函数运算，如矩阵的特征值运算、行列式运算、矩阵的范数运算和矩阵的条件数运算等。掌握这些常用的函数运算，是进行科学运算的基础，本节将介绍一些关于矩阵特征参数的最具有代表性的函数，如表 4-1 所示。熟练地掌握这些函数之后，用户可以更加轻松地进行矩阵方面的运算。

表 4-1　矩阵的特征值函数

函 数 名	功 能 描 述
^	矩阵的乘方运算
sqrtm	矩阵的开方运算
expm	矩阵的指数运算
logm	矩阵的对数运算
cond	求矩阵的条件数
condest	求矩阵的 1 范数估计
condeig	求矩阵和特征值有关的条件数
det	求矩阵的行列式
eig 或是 eigs	求矩阵的特征值和特征向量
eig	特征值矩阵
funm	矩阵的任意函数
gsvd	广义奇异值
inv	矩阵求逆
norm 或是 normest	求矩阵和向量的范数
null	右 0 空间
pinv	伪逆矩阵
poly	求矩阵的特征多项式
polyvalm	求矩阵多项式的值
rank	求矩阵的秩
trace	求矩阵的迹

下面，对几个比较常用的函数来进行详细讲解，用户可以依次来了解其他函数的用法。

1. 矩阵的乘方运算和开方运算

在 MATLAB 7.0 中，可以使用 A^p 来计算 A 的 p 次方，使用 sqrtm 函数来对矩阵进行开方运算，如果有 $X*X=A$，则有 $sqrtm(A)=X$。

例 4-14　计算矩阵的 10 次方。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> A=[1 2 3 4;4 5 6 7;4 5 6 7;8 9 10 11]
A =
      1     2     3     4
      4     5     6     7
      4     5     6     7
      8     9    10    11
>> B=A^10
B =
   1.0e+013 *
     0.7091    0.8608    1.0124    1.1641
     1.4364    1.7436    2.0508    2.3581
     1.4364    1.7436    2.0508    2.3581
     2.4060    2.9207    3.4353    3.9500
>>
```

例 4-15　求矩阵 B 的开方并检验其正确性。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> A=magic(5)
A =
     17    24     1     8    15
     23     5     7    14    16
      4     6    13    20    22
     10    12    19    21     3
     11    18    25     2     9
>> B=sqrtm(A)
B =
  3.5456 + 1.3435i   2.7962 - 1.5926i  -0.3000 + 0.8860i   0.5706 - 0.1082i   1.4499 - 0.5287i
  2.5518 - 1.5247i   2.1700 + 2.6661i   0.6534 + 0.2501i   1.2598 - 0.6915i   1.4272 - 0.7001i
 -0.0313 + 0.5290i   0.6707 + 0.3211i   3.3478 + 1.7354i   1.9336 - 0.9417i   2.1415 - 1.6437i
  0.9124 - 0.1853i   0.9210 - 0.3527i   1.8626 - 0.9590i   4.2785 + 0.5576i   0.0878 + 0.9393i
  1.0838 - 0.1626i   1.5043 - 1.0419i   2.4984 - 1.9125i   0.0198 + 1.1838i   2.9560 + 1.9331i
>> B^2
ans =
 17.0000 + 0.0000i  24.0000 + 0.0000i   1.0000 - 0.0000i   8.0000 - 0.0000i  15.0000 - 0.0000i
 23.0000 + 0.0000i   5.0000 + 0.0000i   7.0000 + 0.0000i  14.0000 + 0.0000i  16.0000 + 0.0000i
```

```
   4.0000 - 0.0000i   6.0000 + 0.0000i  13.0000 + 0.0000i  20.0000 - 0.0000i  22.0000 - 0.0000i
  10.0000 - 0.0000i  12.0000 + 0.0000i  19.0000 + 0.0000i  21.0000 - 0.0000i   3.0000 - 0.0000i
  11.0000 - 0.0000i  18.0000 + 0.0000i  25.0000 - 0.0000i   2.0000 - 0.0000i   9.0000 - 0.0000i
>>
```

由例 4-15 可见，矩阵的开方运算和乘方运算互为逆运算。

2. 矩阵的指数和对数运算

矩阵的指数运算用 expm 函数来实现，*expm(X)* = *V*diag(exp(diag(D)))/V*(其中 X 为已知矩阵，*[V,D] =eig(X)*)。对数运算用 logm 函数来实现，*L* = *logm(A)*，与矩阵的指数运算互为逆运算。

例 4-16　矩阵的指数和对数运算。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> X=rand(4)
X =
    0.9501    0.8913    0.8214    0.9218
    0.2311    0.7621    0.4447    0.7382
    0.6068    0.4565    0.6154    0.1763
    0.4860    0.0185    0.7919    0.4057
>> Y=expm(X)
Y =
    4.5995    3.3360    3.9090    3.4729
    1.6117    3.0514    2.1482    2.1212
    2.0584    1.8664    3.2516    1.5620
    1.8482    1.1587    2.2976    2.5245
>> A=randn(4)
A =
   -0.4326   -1.1465    0.3273   -0.5883
   -1.6656    1.1909    0.1746    2.1832
    0.1253    1.1892   -0.1867   -0.1364
    0.2877   -0.0376    0.7258    0.1139
>> B=logm(A)
B =
   0.9869 + 2.3693i  -0.1483 + 1.1887i   0.6781 - 0.3176i  -0.9346 - 0.8298i
  -0.8132 + 1.3557i   0.6633 + 0.6802i  -0.6967 - 0.1817i   1.1654 - 0.4748i
  -0.8050 - 1.5545i   0.4793 - 0.7799i  -0.7019 + 0.2084i  -1.0269 + 0.5445i
   1.0296 + 0.3319i  -0.1240 + 0.1665i   1.3317 - 0.0445i  -0.2798 - 0.1162i
>>
```

3. 矩阵的逆运算

矩阵的逆运算是矩阵运算中很重要的运算之一，在线性代数的相关教材中有很详细的

论述，矩阵可逆的充分必要条件就是矩阵的行列式不为零。在 MATLAB 7.0 中，所有复杂的理论都被简化成了一个函数 inv。

例 4-17　求矩阵 A 的逆，A 的定义如下边的程序所示。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
A= [1 0 0 0;1 2 0 0;2 1 3 0;1 2 1 4]
>> A=[1 0 0 0;1 2 0 0;2 1 3 0;1 2 1 4]
A =
     1     0     0     0
     1     2     0     0
     2     1     3     0
     1     2     1     4
>> B=inv(A)
B =
    1.0000         0         0         0
   -0.5000    0.5000         0         0
   -0.5000   -0.1667    0.3333         0
    0.1250   -0.2083   -0.0833    0.2500
>>
```

4. 矩阵的行列式运算

当矩阵的行和列相同时，可以进行矩阵的行列式操作，MATLAB 7.0 提供了 det 函数来求矩阵的行列式的值。

例 4-18　求矩阵 A 的及其逆矩阵 B 的行列式的值，其中 A 和 B 的值取上例。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> x=det(A)
x =
    24
>> y=det(B)
y =
    0.0417
>> i=x*y
i =
     1
>>
```

5. 矩阵的特征值运算

在 MATLAB 7.0 中，可以利用 eig 和 eigs 两个函数来进行矩阵的特征值运算，其使用格式如下。

- *E* = *eig*(*X*)命令生成由矩阵 *X* 的特征值所组成的一个列向量。
- [*V*, *D*] = *eig*(*X*)命令生成两个矩阵 *V* 和 *D*，其中 *V* 是以矩阵 *X* 的特征向量作为列向

量组成的矩阵，D 是由矩阵 X 的特征值作为主对角线元素构成的对角矩阵。

- eigs 函数使用迭代法求解矩阵的特征值和特征向量。
- *D* = *eig*(*X*)命令生成由矩阵 X 的特征值所组成的一个列向量。X 必须是方阵，最好是大型稀疏矩阵。
- [*V*, *D*] = *eig*(*X*)命令生成两个矩阵 V 和 D，其中 V 是以矩阵 X 的特征向量作为列向量组成的矩阵，D 是由矩阵 X 的特征值作为主对角线元素构成的对角矩阵。

例 4-19　求矩阵 X 和 A 的特征值和特征向量。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> X=magic(3)
X =
     8     1     6
     3     5     7
     4     9     2
>> A=[1 0 0;0 0 3;0 9 0]
A =
     1     0     0
     0     0     3
     0     9     0
>> E=eig(X)
E =
   15.0000
    4.8990
   -4.8990

>> [V,D]=eig(X)
V =
   -0.5774   -0.8131   -0.3416
   -0.5774    0.4714   -0.4714
   -0.5774    0.3416    0.8131
D =
   15.0000         0         0
         0    4.8990         0
         0         0   -4.8990
>> D=eigs(A)
D =
   -5.1962
    5.1962
    1.0000
>> [V,D]=eigs(A)
V =
         0         0    1.0000
    0.5000    0.5000         0
```

```
    -0.8660    0.8660         0
D =
    -5.1962         0         0
          0    5.1962         0
          0         0    1.0000
>>
```

6. 矩阵(向量)的范数运算

数值分析与计算方法的不同之处在于引入了范数的概念，它分为 2-范数、1-范数、无穷范数和Frobenius 范数等。这些范数反映了矩阵(向量)的某些特性。MATLAB 7.0 中用norm 函数和 normest 函数来计算矩阵(向量)的范数。其使用格式如下。

norm 函数

- *norm* (*X*)用来计算矩阵(向量)*X* 的 2-范数。
- *norm* (*X*, 2)与 *norm* (*X*)的功能相同。
- *norm* (*X*, 1)用来计算矩阵(向量)*X* 的 1-范数。
- *norm* (*X*, inf)用来计算矩阵(向量)*X* 的无穷范数。
- *norm* (*X*, '*fro*')用来计算矩阵(向量)*X* 的 Frobenius 范数。

normest 函数

- normest 函数只能计算矩阵(向量)的 2-范数。并且是其 2-范数的估计值。适用于计算 *norm* (*X*)比较费时的情况，格式为 *normest*(*X*)。

例 4-20　矩阵范数的计算。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> X=hilb(4)
X =
    1.0000    0.5000    0.3333    0.2500
    0.5000    0.3333    0.2500    0.2000
    0.3333    0.2500    0.2000    0.1667
    0.2500    0.2000    0.1667    0.1429
>> norm(4)
ans =
     4
>> norm(X)
ans =
1.5002
>> norm(X,2)
ans =
    1.5002
>> norm(X,1)
ans =
    2.0833
>> norm(X,inf)
```

```
ans =
      2.0833
>> norm(X,'fro')
ans =
      1.5097
>> normest(X)
ans =
      1.5002
>>
```

7. 矩阵的条件数运算

矩阵的条件数是判断矩阵“病态”程度的一个量值，矩阵 A 的条件数越大，表明 A 越“病态”，反之，表明 A 越“良态”。例前面介绍的 Hilbert 矩阵就是有名的病态矩阵。MATLAB 7.0 提供了 3 个函数来求矩阵的条件数，它们是 cond、rcond 和 condest。其使用格式如下。

cond 函数

- cond 函数用于计算矩阵的条件数，*cond* (*X*)返回关于矩阵 *X* 的 2-范数的条件数，
- *cond* (*X*, *P*)关于矩阵 *X* 的 P-范数的条件数(P 为 1、2、inf 或 fro)。

rcond 函数

- rcond 函数用于计算矩阵条件数的倒数值，所以当矩阵 *X*“病态”时，*rcond* (*X*)就接近 0，*X*“良态”时，*rcond* (*X*)就接近 1。

condest 函数

- *condest* (*X*)用于计算关于矩阵 *X* 的 1-范数的条件数的估计值。

例 4-21　求魔术矩阵和 Hilbert 矩阵的条件数。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> M=magic(3);
>> H=hilb(4);
>> c1=cond(M)
c1 =
      4.3301
>> c2=cond(M,1)
c2 =
      5.3333
>> c3=rcond(M)
c3
      0.1875
>> c4=condest(M)
c4 =
      5.3333
>> h1=cond(H)
h1 =
```

```
  1.5514e+004
>> h2=cond(H,inf)
h2 =
  2.8375e+004
>> h3=rcond(H)
h3 =
  3.5242e-005
>> h4=condest(H)
h4 =
  2.8375e+004
>>
```

从例 4-21 可以看出，魔术矩阵比较“良态”，而 Hilbert 矩阵是“病态”的。

8. 矩阵的秩

由线性代数知识可知，秩的是线性代数中相当重要的概念之一，通常矩阵都可以经过初等行或列变换，将其转化为行阶梯形矩阵，而行阶梯形矩阵所包含非零行的行数是一定的，这个确定的非零行的行数就是矩阵的秩。

在 MATLAB 7.0 中，矩阵的秩可以通过函数 rank 来求得。

例 4-22　求矩阵的秩。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> T=rand(6)
T =
    0.9501    0.4565    0.9218    0.4103    0.1389    0.0153
    0.2311    0.0185    0.7382    0.8936    0.2028    0.7468
    0.6068    0.8214    0.1763    0.0579    0.1987    0.4451
    0.4860    0.4447    0.4057    0.3529    0.6038    0.9318
    0.8913    0.6154    0.9355    0.8132    0.2722    0.4660
    0.7621    0.7919    0.9169    0.0099    0.1988    0.4186
>> r=rank(T)
r =
6
```

由上计算结果可知，矩阵 T 为满秩矩阵。

```
>> T1=[1 1 1;2 2 3]
T1 =
     1     1     1
     2     2     3
>> r=rank(T1)
r =
     2
>>
```

由上计算结果可知，矩阵 T1 为行满秩矩阵。

9. 矩阵的迹

矩阵的迹是指矩阵主对角线上所有元素的和，也是矩阵各特征值。

在 MATLAB 7.0 中，矩阵的迹可以通过 trace 函数求得。

例 4-23　求矩阵的迹，并与矩阵特征值进行验证。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> M=magic(5)
M =
    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9
>> T=trace(M)
T =
    65
>> T1=eig(M)
T1 =
   65.0000
  -21.2768
  -13.1263
   21.2768
   13.1263
>> T2=65-21-13+21+13
T2 =
    65
>>
```

4.2.4　矩阵的分解运算

MATLAB 7.0 的数学处理能力之所以强大，很大一部分的原因就是它的矩阵函数功能的扩展，本小节主要介绍矩阵函数中的矩阵分解运算。矩阵分解在数值分析科学中占有重要的地位，常用的分解方法有三角分解(lu)、正交分解(qr)、特征值分解(eig)、奇异值分解(svd)和 Chollesky 分解(chol)等。

1. 三角分解(lu)

lu 分解是矩阵分解的最基本的方法，它在线性方程的直接解法中有重要的应用。由数值分析知识可知，非奇异矩阵 A(n×n)，如果其顺序主子式均不为零，则存在惟一的单位下三角 L 和上三角阵 U，从而使得 $A = L * U$。

在 MATLAB 7.0 中，lu 分解可以通过 lu 函数实现。常见的具体应用形式有 3 种，下面分别予以介绍。

- $[L,U]=lu(X)$ ：产生一个上三角矩阵 U 和一个下三角矩阵 L，使得矩阵 L 和矩阵 U 满足关系式 $X=L*U$， X 可以不为方阵。
- $[L,U,P]=lu(X)$：产生一个单位下三角矩阵 L、一个上三角矩阵 U 和交换矩阵 P，使得这 3 个矩阵满足关系式 $P*X=L*U$。
- $Y=lu(X)$ 命令中：如果 X 是满矩阵，将产生一个 lapack's 的 dgetrf 或 zgetrf 的输出常式矩阵 Y；如果 X 是稀疏矩阵，产生的矩阵 Y 将包含严格的下三角矩阵 L 和上三角矩阵 U。在这两种情况下，都不会有交换矩阵 P。

例 4-24　求矩阵 X 三角分解后的矩阵。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> X=[6 2 1 -1;2 4 1 0;1 1 4 -1;-1 0 -1 3]
X =
     6     2     1    -1
     2     4     1     0
     1     1     4    -1
    -1     0    -1     3
>> [L,U]=lu(X)
L =
    1.0000         0         0         0
    0.3333    1.0000         0         0
    0.1667    0.2000    1.0000         0
   -0.1667    0.1000   -0.2432    1.0000
U =
    6.0000    2.0000    1.0000   -1.0000
         0    3.3333    0.6667    0.3333
         0         0    3.7000   -0.9000
         0         0         0    2.5811
>> [L,U,P]=lu(X)
L =
    1.0000         0         0         0
    0.3333    1.0000         0         0
    0.1667    0.2000    1.0000         0
   -0.1667    0.1000   -0.2432    1.0000
U =
    6.0000    2.0000    1.0000   -1.0000
         0    3.3333    0.6667    0.3333
         0         0    3.7000   -0.9000
         0         0         0    2.5811
P =
     1     0     0     0
     0     1     0     0
```

```
       0       0       1       0
       0       0       0       1
>> Y=lu(X)
Y =
    6.0000    2.0000    1.0000   -1.0000
    0.3333    3.3333    0.6667    0.3333
    0.1667    0.2000    3.7000   -0.9000
   -0.1667    0.1000   -0.2432    2.5811
>>
```

2. 正交分解(qr)

在数值分析中，为了求解矩阵的特征值，引入了一种矩阵的分解方法，即 qr 法。即对于矩阵 A(n×n)，如果 A 非奇异，则存在正交矩阵 Q 和上三角矩阵 R，使得 A 满足关系式 $A=Q*R$，并且当 R 的对交元都为正时，qr 分解是惟一的。

在 MATLAB 7.0 中，矩阵的 qr 分解可以由 qr 函数实现。常见的具体应用形式有 4 种，分别介绍如下。

- $[Q,R]=qr(A)$：产生一个与 A 维数相同的上三角矩阵 R 和一个正交矩阵 Q，使得满足关系式 $A=Q*R$。
- $[Q,R,E]=qr(A)$：产生一个交换矩阵 E、一个上三角矩阵 R 和正交阵 Q，这 3 个矩阵满足关系式 $A*E=Q*R$。
- $[Q,R]=qr(A,0)$：对矩阵 A 进行有选择的 qr 分解。当矩阵 A 为 m×n 并且 m>n，那么只会产生具有前 n 列的正交矩阵 Q。
- $R=qr(A)$：只产生矩阵 R，并且满足 $R=chol(A'*A)$。

例 4-25　求矩阵 A 的正交分解。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> A=[17 3 4;3 1 12;4 12 8]
A =
    17     3     4
     3     1    12
     4    12     8
>> [Q,R]=qr(A)
Q =
   -0.9594    0.2294    0.1643
   -0.1693   -0.0023   -0.9856
   -0.2257   -0.9733    0.0411
R =
  -17.7200   -5.7562   -7.6749
         0  -10.9939   -6.8967
         0         0  -10.8412
>> [Q,R,E]=qr(A)
Q =
```

```
    -0.9594    0.2617   -0.1054
    -0.1693   -0.8328   -0.5270
    -0.2257   -0.4878    0.8433
R =
  -17.7200   -7.6749   -5.7562
         0  -12.8490   -5.9010
         0         0    9.2760
E =
     1     0     0
     0     0     1
     0     1     0
>> [Q,R]=qr(A,0)
Q =
    -0.9594    0.2294    0.1643
    -0.1693   -0.0023   -0.9856
    -0.2257   -0.9733    0.0411
R =
  -17.7200   -5.7562   -7.6749
         0  -10.9939   -6.8967
         0         0  -10.8412
>> R=qr(A)
R =
  -17.7200   -5.7562   -7.6749
    0.0864  -10.9939   -6.8967
    0.1152    0.9781  -10.8412
>>
```

3. 特征值分解(eig)

矩阵的特征值分解在 MATLAB 7.0 中，也是利用函数 eig，但是为了分解，需要在形式上作一定的变化。其使用格式如下。

- $[V,D]=eig(X)$ 命令生成两个矩阵 V 和 D，其中 V 是以矩阵 X 的特征向量作为列向量组成的矩阵，D 是由矩阵 X 的特征值作为主对角线元素构成的对角矩阵，使得满足关系式 $X*V=V*D$。
- $[V,D]=eig(A,B)$ 命令对矩阵 A、B 做广义特征值分解，使得满足关系式 $A*V=B*V*D$。

例 4-26　求矩阵的特征值分解。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> A=magic(4)
A =
    16     2     3    13
     5    11    10     8
     9     7     6    12
```

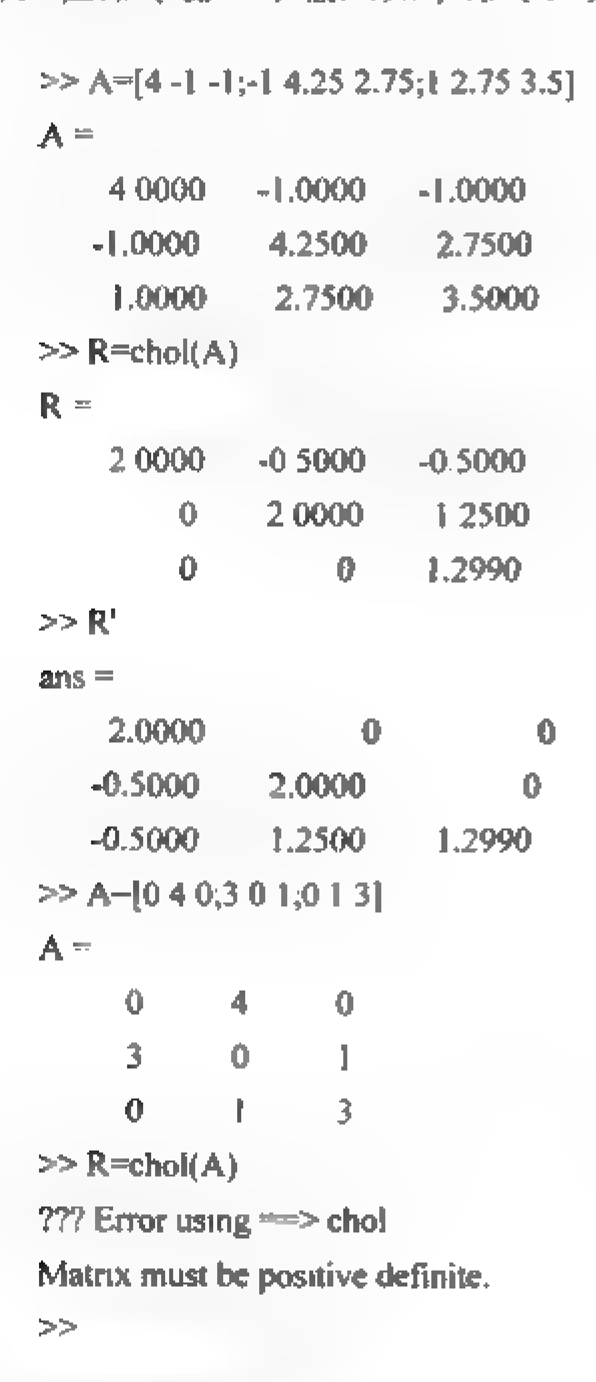
```
     4    14    15     1
>> [V,D]=eig(A)
V =
   -0.5000   -0.8236    0.3764   -0.2236
   -0.5000    0.4236    0.0236   -0.6708
   -0.5000    0.0236    0.4236    0.6708
   -0.5000    0.3764   -0.8236    0.2236
D =
   34.0000         0         0         0
         0    8.9443         0         0
         0         0   -8.9443         0
         0         0         0    0.0000
>> Z=A*V-V*D
Z =
  1.0e-013 *
   -0.1066    0.0711   -0.0222   -0.0154
   -0.1776    0.0577   -0.0105   -0.0264
   -0.1066    0.0247   -0.0178   -0.0380
    0.0711    0.0799         0   -0.0154
>> B=[17 3 4 2;3 1 12 6;4 12 8 7;1 2 3 4]
B =
    17     3     4     2
     3     1    12     6
     4    12     8     7
     1     2     3     4
>> [V,D]=eig(A,B)
V =
   -0.0517    0.8287    1.0000   -0.3333
   -0.3590    0.2175    0.2859   -1.0000
   -0.4474    0.0914   -0.5660    1.0000
    1.0000    1.0000   -0.7016    0.3333
D =
   -5.7955         0         0         0
         0    1.5765         0         0
         0         0    0.4054         0
         0         0         0   -0.0000
>> Z=A*V-B*V*D
Z =
  1.0e-013 *
   -0.1776    0.1066   -0.0799    0.0372
    0.1177    0.0355    0.1243   -0.0228
   -0.0089    0.0711    0.1232   -0.1031
    0.0888    0.1243    0.0600    0.0047
>>
```

4. Chollesky 分解(chol)

由数值分析的知识可知，当矩阵 A(n×n)对称正定时，则存在唯一的对角元素为正的上三角矩阵 R，使得 $A=R*R'$，这种分解就称为 Chollesky 分解，当限定 R 的对角元素为正时，这种分解是唯一的。在 MATLAB 7.0 中，这种分解可以通过函数 chol 实现，需要注意的是当矩阵 A 为非正定阵时，MATLAB 7.0 将会提示出错误信息。

例 4-27　求矩阵的 Chollesky 分解。

解：在命令窗口中输入如下命令，并按 Enter 键确认。



```
>> A=[4 -1 -1;-1 4.25 2.75;1 2.75 3.5]
A =
    4.0000   -1.0000   -1.0000
   -1.0000    4.2500    2.7500
    1.0000    2.7500    3.5000
>> R=chol(A)
R =
    2.0000   -0.5000   -0.5000
         0    2.0000    1.2500
         0         0    1.2990
>> R'
ans =
    2.0000         0         0
   -0.5000    2.0000         0
   -0.5000    1.2500    1.2990
>> A=[0 4 0;3 0 1;0 1 3]
A =
     0     4     0
     3     0     1
     0     1     3
>> R=chol(A)
??? Error using ==> chol
Matrix must be positive definite.
>>
```

5. 奇异值分解(svd)

在 MATLAB 7.0 中，矩阵的奇异值分解可以通过 svd 函数实现。

具体应用形式有两种，下面分别予以介绍。

- $[U,S,V]=svd(X)$ 命令产生一个与矩阵 X 维数相同的对角矩阵 S、正交矩阵 U 和正交矩阵 V，并且使得这 4 个矩阵满足关系式 $X=U*S*V'$。
- $[U,S,V]=svd(X,0)$ 命令进行奇异值的最佳分解。X 为 M×N 阶矩阵，当 M>N 时，生成的矩阵 U 只有前 N 列元素被计算出来，并且 S 为 N×N 阶矩阵。

例 4-28　求矩阵的奇异值分解。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> X=[1 2 3 ;4 5 6;7 8 9]
X =
     1     2     3
     4     5     6
     7     8     9
>> [U,S,V]=svd(X)
U =
   -0.2148    0.8872    0.4082
   -0.5206    0.2496   -0.8165
   -0.8263   -0.3879    0.4082
S =
   16.8481         0         0
         0    1.0684         0
         0         0    0.0000
V =
   -0.4797   -0.7767   -0.4082
   -0.5724   -0.0757    0.8165
   -0.6651    0.6253   -0.4082
>> X=[1 2 3 ;4 5 6;7 8 9;10 11 12]
X =
     1     2     3
     4     5     6
     7     8     9
    10    11    12
>> [U,S,V]=svd(X,0)
U =
   -0.1409    0.8247    0.5456
   -0.3439    0.4263   -0.6919
   -0.5470    0.0278   -0.2531
   -0.7501   -0.3706    0.3994
S =
   25.4624         0         0
         0    1.2907         0
         0         0    0.0000
V =
   -0.5045   -0.7608   -0.4082
   -0.5745   -0.0571    0.8165
   -0.6445    0.6465   -0.4082
>>
```

4.2.5　矩阵的一些特殊处理函数

1. 矩阵的变维

在 MATLAB 7.0 中，使用 reshape 函数来对矩阵进行变维操作，其使用格式如下。

- $reshape(X,M,N)$ 命令将矩阵 X 的所有元素分配到一个 $M\times N$ 的新矩阵，当矩阵 X 的元素不是 $M\times N$ 时，将返回一个错误。
- $reshape(X,M,N,p,...)$ 命令返回由矩阵 X 的元素组成的 $M\times N\times P\times......$ 多维矩阵，如果 $M\times N\times P\times......$ 与 X 的元素数不一样时，将返回错误。
- $reshape(X,[M,N,p,...])$ 命令与 $reshape(X,M,N,p,...)$ 命令的效果相同。

例 4-29　使用 reshape 函数进行矩阵的变维运算。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> A=rand(4,2)
A =
     0.6154     0.1763
     0.7919     0.4057
     0.9218     0.9355
     0.7382     0.9169
>> reshape(A,2,4)
ans =
     0.6154     0.9218     0.1763     0.9355
     0.7919     0.7382     0.4057     0.9169
>> reshape(A,[2 2 2])
ans(:,:,1) =
     0.6154     0.9218
     0.7919     0.7382
ans(:,:,2) =
     0.1763     0.9355
     0.4057     0.9169
>>
```

2. 矩阵的变向

矩阵的变向包括对矩阵进行旋转、上下翻转、左右翻转以及对指定的维进行翻转。分别由 rot90、flipud、fliplr、和 flipdim 函数来实现。其用法如下。

- $rot90(A)$ 命令返回矩阵 A 按逆时钟方向旋转 90 度所得的矩阵。
- $rot90(A,K)$ 命令返回矩阵 A 按逆时钟方向旋转 $90*K$ 度所得的矩阵 ($K=\pm1,\pm2,.....$)。
- $filpud(X)$ 命令将矩阵 X 上下翻转。

- $filplr(X)$ 命令将矩阵 X 左右翻转。
- $filp\dim(X,DIM)$ 命令将矩阵 X 的第 DIM 维翻转。

例 4-30　矩阵的变向操作。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> X=[1 4;2 5;3 6]
X =
     1     4
     2     5
     3     6
>> rot90(X)
ans =
     4     5     6
     1     2     3
>> rot90(X,-1)
ans =
     3     2     1
     6     5     4
>> flipud(X)
ans =
     3     6
     2     5
     1     4
>> fliplr(X)
ans =
     4     1
     5     2
     6     3
>> flipdim(X,2)
ans =
     4     1
     5     2
     6     3
>>
```

4.2.6　特殊矩阵的生成

在介绍了矩阵的生成之后，本书再介绍一些特殊矩阵的生成，让大家对各种不同的矩阵都有所了解，如表 4-2 所示。

表 4-2　特殊矩阵生成函数

函　　数	函数对应的功能
[ ]	生成空矩阵
zeros	生成 0 矩阵
eye	生成单位矩阵
ones	生成全 1 矩阵
tril 和 triu	生成上下三角矩阵
diag	生成对角矩阵
gallery	生成一些小的测试矩阵
hadamard	生成 Hadamard 矩阵
hankel	生成 Hankel 矩阵
hilb	生成 Hilbert 矩阵
invhilb	生成反 Hilbert 矩阵
magic	生成魔术矩阵
pascal	生成 n 阶 Pascal 矩阵
rand	生成服从 0－1 分布的随机矩阵
randn	生成服从正态分布的随机矩阵
rosser	典型的对称矩阵特征值的问题测试
toeplitz	生成 Toeplitz 矩阵
vander	生成范德蒙矩阵
wilkinson	生成 Wilkinson 矩阵
compan	生成多项式的伴随矩阵

下面，对比较常用的函数来进行详细讲解，用户可以依次来了解其他函数的用法。

1. 零矩阵和全 1 矩阵的生成

零矩阵指的是各个元素都为零的矩阵，在 MATLAB 7.0 语言中，使用 zeros 函数来生成一个零矩阵，它的使用格式如下。

- $A = zeros(M, N)$ 命令中，A 为生成的零矩阵，M 和 N 分别为生成矩阵的行和列。
- 如果存在已知矩阵 B，要生成与 B 维数相同的矩阵，可以使用命令 $A = zeros(size(B))$。
- 当要生成一个方阵时，也可以直接使用命令 $A = zeros(N)$，此时 MATLAB 7.0 将生成一个 N 阶方阵。而全 1 矩阵的生成与零矩阵的生成类似，只是使用 ones 函数来实现。

例 4-31　零矩阵的生成。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> A=zeros(4,5)
A =
```

```
        0     0     0     0     0
        0     0     0     0     0
        0     0     0     0     0
        0     0     0     0     0
>> B=[1 2 3 4 5;2 3 4 5 6,9 8 7 6 5;8 7 6 5 4]
B =
        1     2     3     4     5
        2     3     4     5     6
        9     8     7     6     5
        8     7     6     5     4
>> A=zeros(size(B))
A =
        0     0     0     0     0
        0     0     0     0     0
        0     0     0     0     0
        0     0     0     0     0
>> A=zeros(5)
A =
        0     0     0     0     0
        0     0     0     0     0
        0     0     0     0     0
        0     0     0     0     0
        0     0     0     0     0
>> C=ones(5,6)
C =
        1     1     1     1     1     1
        1     1     1     1     1     1
        1     1     1     1     1     1
        1     1     1     1     1     1
        1     1     1     1     1     1
>> C=ones(3)
C=
        1     1     1
        1     1     1
        1     1     1
>>
```

2. 对角矩阵的生成

对角矩阵指的是对角线上的元素为任意数，其他元素为零的矩阵。在 MATLAB 7.0 中，diag 函数来生成一个零矩阵(数学上使用 diag 来表示对角阵)，其使用格式如下。

- $A = diag(V, K)$ 命令中，V 为某个向量，K 为向量 V 偏离主对角线的列数，K 等于零时表示 V 为主对角线，K 为大于零的数时表示 V 在主对角线以上，K 小于零表示 V 在主对角线以下。
- 而 $A = diag(V)$ 相当于 $A = diag(V,0)$。

例 4-32　对角矩阵的生成。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> v=[1 9 8 1 6];
>> diag(v,1)
ans =
     0     1     0     0     0     0
     0     0     9     0     0     0
     0     0     0     8     0     0
     0     0     0     0     1     0
     0     0     0     0     0     6
     0     0     0     0     0     0
>> diag(v,-1)
ans =
     0     0     0     0     0     0
     1     0     0     0     0     0
     0     9     0     0     0     0
     0     0     8     0     0     0
     0     0     0     1     0     0
     0     0     0     0     6     0
>> diag(v)
ans =
     1     0     0     0     0
     0     9     0     0     0
     0     0     8     0     0
     0     0     0     1     0
     0     0     0     0     6
>>
```

3. 随机矩阵的生成

随机矩阵是指矩阵元素是由随机数构成的矩阵。在 MATLAB 7.0 语言中，使用 rand 函数和 randn 函数可以生成多种随机矩阵，其使用格式如下。

- $rand(N)$ 命令生成 $N\times N$ 阶随机矩阵，生成矩阵的元素值在区间(0.0，1.0)之间。
- $rand(M,N)$ 命令生成 $M\times N$ 阶随机矩阵，生成矩阵的元素值在区间(0.0，1.0)之间。
- $randn(N)$ 命令生成 $N\times N$ 阶随机矩阵，生成矩阵的元素值在服从正态分布 N(0,1)。
- $randn(M,N)$ 命令生成 $M\times N$ 阶随机矩阵，生成矩阵的元素值在服从正态分布 N(0,1)。

例 4-33　随机矩阵的生成。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> rand(5)
ans =
    0.9355    0.3529    0.1987    0.7468    0.8462
    0.9169    0.8132    0.6038    0.4451    0.5252
    0.4103    0.0099    0.2722    0.9318    0.2026
    0.8936    0.1389    0.1988    0.4660    0.6721
    0.0579    0.2028    0.0153    0.4186    0.8381
>> randn(5)
ans =
   -0.4326    1.1909   -0.1867    0.1139    0.2944
   -1.6656    1.1892    0.7258    1.0668   -1.3362
    0.1253   -0.0376   -0.5883    0.0593    0.7143
    0.2877    0.3273    2.1832   -0.0956    1.6236
   -1.1465    0.1746   -0.1364   -0.8323   -0.6918
>>
```

4. 范德蒙德矩阵的生成

范德蒙德矩阵是线性代数中一个很重要的矩阵。在 MATLAB 7.0 语言中，使用 vander 函数生成范德蒙德矩阵。其使用格式如下。

- $A = vander(V)$，其中有 $A(i,j) = v(i)^{n-j}$。

例 4-34　范德蒙德矩阵的生成。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> v=[1 3 5 7 9];
>> A=vander(v)
A =
           1           1           1           1           1
          81          27           9           3           1
         625         125          25           5           1
        2401         343          49           7           1
        6561         729          81           9           1
>>
```

5. 魔术矩阵的生成

魔术矩阵是一个人们经常会遇到的矩阵，它是一个方阵，并且方阵的每一行、每一列以及每条主对角线的元素之和都相同(2 阶方阵除外)，在 MATLAB 7.0 语言中，使用 magic 函数来生成魔术矩阵。其使用格式如下。

- $magic(N)$ 命令生成 $N \times N$ 阶的魔术矩阵，使矩阵的每一行、每一列以及每条主对角线的元素和相等。$N > 0$ 且 $N = 2$ 除外。

例 4-35　魔术矩阵的生成。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> magic(2)
ans =
       1     3
       4     2
>> magic(3)
ans =
       8     1     6
       3     5     7
       4     9     2
>> magic(4)
ans =
      16     2     3    13
       5    11    10     8
       9     7     6    12
       4    14    15     1
>>
```

6. Hilbert 矩阵和反 Hilbert 矩阵的生成

Hilbert 矩阵是有名的病态矩阵，它的第 i 行、第 j 列的元素值为 1/(i+j-1)。在 MATLAB 7.0 中，使用 hilb 函数来生成 Hilbert 矩阵。而反 Hilbert 矩阵是 Hilbert 矩阵的逆矩阵，在 MATLAB 7.0 语言中，使用 invhilb 函数来生成反 Hilbert 矩阵。它们的使用格式如下。

- $hilb(N)$)命令生成 $N \times N$ 阶 Hilbert 矩阵。
- $invhilb(N)$ 命令生成 $N \times N$ 阶反 Hilbert 矩阵。

例 4-36　Hilbert 矩阵和反 Hilbert 矩阵的生成。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> A=hilb(5)
A =
    1.0000    0.5000    0.3333    0.2500    0.2000
    0.5000    0.3333    0.2500    0.2000    0.1667
    0.3333    0.2500    0.2000    0.1667    0.1429
    0.2500    0.2000    0.1667    0.1429    0.1250
    0.2000    0.1667    0.1429    0.1250    0.1111
>> B=invhilb(5)
B =
          25        -300        1050       -1400         630
        -300        4800      -18900       26880      -12600
        1050      -18900       79380     -117600       56700
       -1400       26880     -117600      179200      -88200
         630      -12600       56700      -88200       44100
>> C=A*B
C =
    1.0000         0         0         0         0
         0    1.0000         0         0         0
```

```
         0         0    1.0000   -0.0000         0
         0         0         0    1.0000         0
         0         0         0         0    1.0000
>>
```

上面介绍了一些比较常见的特殊矩阵，还有许多其他形式的特殊矩阵，这里不再一一说明，有兴趣的读者可以参考有关资料和 MATLAB 7.0 自带的帮助系统。

4.3　数组及其运算

由数学知识可知，数组和矩阵有着不同的概念。在 MATLAB 7.0 中，数组和矩阵在形式上有很多一致性，但是实际上它们遵循不同的运算规则。对于初学者来说，容易将这两者混为一谈，从而将两者的语法混淆，导致语法错误，使程序无法正常运行，更严重的是，混淆两者容易导致一些隐蔽的错误，虽然程序能够通过编译，但是程序所表达的意思与编写者的意思已经发生了变化，从而导致错误的结果。因此，本节有必要对数组及其运算做出简要介绍，使用户尽量避免这样的错误。

4.3.1　数组寻址和排序

1. 数组寻址

由于数组是由多个元素组成的，因此，在访问数组中的单个或是多个元素时，有必要对数组进行寻址运算。在 MATLAB 7.0 中，通过对数组下表的访问来实现数组寻址。

例 4-37　数组的寻址运算。

首先，生成一个 1×10 的数组 A。

```
>> A=randn(1,10)
A =   -0.4326    -1.6656     0.1253     0.2877    -1.1465     1.1909     1.1892    -0.0376
0.3273     0.1746
>>
```

访问 A 的单个元素时，可以直接采用访问下表的方法，例如，要访问 A 的第 4 个元素，可以进行如下操作。

```
>> A(4)
ans =
    0.2877
>>
```

如果用户需要一次访问一块数据，可以使用 MATLAB 7.0 提供的冒号。如要访问 A 中的第 2 到第 6 个元素，可以采用如下格式。

```
>> A(2:6)
ans =
    1.6656    0.1253    0.2877   -1.1465    1.1909
>> A(6:-2:1)
ans =
    1.1909    0.2877   -1.6656
>>
```

可以看出，使用冒号只能访问数组中的连续元素，如要访问多个不连续的元素，可以使用中括号进行操作。如要访问 A 中 1、3、7 和 4 号元素，可以使用如下操作。

```
>> A([1 3 7 4])
ans =
   -0.4326    0.1253    1.1892    0.2877
>>
```

此外，MATLAB 7.0 还提供了 end 参数来表示数组的结尾，如下所示。

```
>> A(4:end)
ans =
    0.2877   -1.1465    1.1909    1.1892   -0.0376    0.3273    0.1746
>>
```

2. 数组排序

对一个任意给定的数组，其数组元素往往是没有规律性的，在实际应用中，往往需要对数组元素进行排序，在 MATLAB 7.0 中，使用 sort 函数对数组进行排序，其使用格式如下。

- $sort(X)$ 命令将数组 X 中的元素按升序排列。
- 当 X 是多维数组时，$sort(X)$ 命令将 X 中的各列元素按升序排列。
- 当 X 是一个字符型单元数组，那么 $sort(X)$ 命令将 X 中的元素按 ASCII 码进行升序排序。
- $Y = sort(X, DIM, MODE)$ 命令中增加了两个参数，DIM 选择用于排列的维，而 $MODE$ 决定了排序的方式，选择'ascend'将按升序排列，而选择'descend'将按降序排列。该命令生成的数组 Y 与 X 有相同的数组结构和型号。
- 当 X 的数据类型为复数时，将按各元素的模 $abs(X)$ 进行排序。

例 4-38　使用 sort 函数进行数组排序。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>>  X=[3 7 5
       0 4 2]
X =

     3     7     5
```

```
         0     4     2

>> sort(X,1)
ans =
     0     4     2
     3     7     5
>> sort(2)
ans =
     2
>> sort(X,2)
ans =
     3     5     7
     0     2     4
>>
```

4.3.2　数组的基本数值运算

简单地说来，MATLAB 7.0 数组运算符由矩阵运算符前面增加一点“.”来表示，例如“.*”、“.\”和“.^”等。

1. 数组的加法(减法)

数组的加法(减法)运算与矩阵的加法(减法)运算相同，因此运算符号“+”(“-”)既可以被矩阵接受，也可以为数组接受。

例 4-39　计算数组 $X=[1,4,7]$ 和 $Y=[2,5,8]$ 的和与差。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> X=[1 4 7];
>> Y=[2 5 8];
>> Z=X-Y
Z =
    -1    -1    -1
>> V=X+Y
V =
     3     9    15
>>
```

2. 数组的乘法(除法)

数组的乘法(除法)运算在 MATLAB 7.0 中用符号“.*”(“./”)表示。如果数组 X 和数组 Y 具有相同的维数，则数组的乘法运算 $X.*Y$ 表示 X 和 Y 中单个元素之间的对应乘积，需要注意，X 和 Y 要有相同的维数，这样数组的乘法才有意义。对于数组的除法运算，与乘法类似。

例 4-40　数组的乘法和除法运算。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> X=[10 52 96 12 56]
X =
     10     52     96     12     56
>> Y=[2 26 3 4 8]
Y =
      2     26      3      4      8
>> Z=[10 52 96 12 56 42]
Z =
     10     52     96     12     56     42
>> Z1=X.*Y
Z1 =
          20          1352          288          48          448
>> Z2=X.*Z
??? Error using ==> .*
Matrix dimensions must agree.
>> Z3=X./Y
Z3 =
      5      2     32      3      7
>> Z4=X.\Y
Z4 =
     0.2000     0.5000     0.0313     0.3333     0.1429
>> Z5=X./Z
??? Error using ==> ./
Matrix dimensions must agree.
>>
```

由例 4-40 的各运算步骤可以看出：数组乘法运算必须要符合维数要求，否则 MATLAB 7.0 会提示出错误信息；数组的除法运算也必须满足维数要求，同时，读者还要注意“./”和“.\”计算结果是完全不一样的。

3. 数组的乘方

数组的乘方运算在 MATLAB 7.0 中用符号“.^”表示。数组的乘方运算有 3 种不同的形式。下面分别予以介绍。

- 两个数组之间乘方运算的情况。

例 4-41　计算数组 $X=[1,4,7]$ 和 $Y=[2,5,8]$ 乘方。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> X=[1 4 7]
X =
      1      4      7
>> Y=[2 5 8]
Y =
```

```
      2       5       8
>> Z=X.^Y
Z =
              1          1024        5764801
```

◆ 1 个数组的某个具体数值的乘方，即计算数组乘方运算时指数为标量的情况。

例 4-42　计算数组 $X=[3,6,9]$ 的 3 次幂。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> X=[3 6 9]
X =
      3       6       9
>> Z=X.^3
Z =
     27     216     729
>>
```

◆ 1 个数组为指数，底数为标量的情况。

例 4-43　计算以 3 为底、数组 X=[4 5 6 7 8 9]为指数的乘方运算。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> X=[4 5 6 7 8]
X =
      4       5       6       7       8
>> Z=3.^X
Z =
             81           243           729          2187          6561
>>
```

4.3.3　数组的关系运算

由数学知识可以知道，两个数通常可以用 6 种关系来进行描述：小于(<)、小于等于(<=)、大于(>)、大于等于(>=)、等于(==)和不等于(~=)。比较两个元素的大小时，如果结果为 1，则表明关系式为真；如果结果为 0，则表明关系式为假。例如关系式 4+3<=6(数学语言表示 4 与 3 的和小于等于 6)，通过上面的叙述可知，此关系式的结果为 0，标明关系式为假。

在 MATLAB 7.0 中，可以通过关系运算符方便地实现数组的关系运算。读者可以观察例 4-41，其中 rem 为 MATLAB 7.0 中的求余函数，完整表达式为 rem(X,n)，X 为被除数，n 为除数。

例 4-44　数组的关系运算。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> M=magic(7)
M
    30    39    48     1    10    19    28
    38    47     7     9    18    27    29
    46     6     8    17    26    35    37
     5    14    16    25    34    36    45
    13    15    24    33    42    44     4
    21    23    32    41    43     3    12
    22    31    40    49     2    11    20
>> N=(rem(M,3))
N =
     0     0     0     1     1     1     1
     2     2     1     0     0     0     2
     1     0     2     2     2     2     1
     2     2     1     1     1     0     0
     1     0     0     0     0     2     1
     0     2     2     2     1     0     0
     1     1     1     1     2     2     2
>> N=(rem(M,3)<=1)
N =
     1     1     1     1     1     1     1
     0     0     1     1     1     1     0
     1     1     0     0     0     0     1
     0     0     1     1     1     1     1
     1     1     1     1     1     0     1
     1     0     0     0     1     1     1
     1     1     1     1     0     0     0
>> N=(rem(M,3)==1)
N =
     0     0     0     1     1     1     1
     0     0     1     0     0     0     0
     1     0     0     0     0     0     1
     0     0     1     1     1     0     0
     1     0     0     0     0     0     1
     0     0     0     0     1     0     0
     1     1     1     1     0     0     0
>> N=(rem(M,3)>=1)
N =
     0     0     0     1     1     1     1
     1     1     1     0     0     0     1
     1     0     1     1     1     1     1
     1     1     1     1     1     0     0
     1     0     0     0     0     1     1
     0     1     1     1     1     0     0
```

```
     1    1    1    1    1    1    1
>>
```

4.3.4 数组的逻辑运算

在各种逻辑运算中，有 3 种逻辑运算：与(&)、或(|)和非(~)。“&”和“|”操作符号可以比较两个标量或者两个通解数组(或矩阵)；对于逻辑非“~”是一个一元操作符。但是对于数组(矩阵)，逻辑运算是针对于数组(矩阵)中的每一个元素。同样，当逻辑为真时，返回值为 1；当逻辑为假时，返回值为 0。

在 MATLAB 7.0 中，逻辑运算通常可以用来生成只含有元素 0 和 1 的矩阵。

4.4 稀疏型矩阵

上面的章节已对通常的 MATLAB 7.0 矩阵做了一些简要的介绍，当创建一个普通的矩阵时，MATLAB 7.0 系统会为矩阵中的每一个元素分配内存。例如，函数 $A = eye(10)$ 创建的矩阵中将有 100 个元素，其主对角线上的元素都为 1，别的元素都为 0，也就是说，矩阵中有 90 个元素都为 0。而这个矩阵要求 100 个单元，但是只有 10 个数据是非 0 的，这就是稀疏矩阵的一个简单例子。总之，稀疏矩阵就是指在矩阵中绝大部分元素为 0 的矩阵。

对于这种情况，MATLAB 7.0 提供了一种更为高级的存储方式，即稀疏矩阵方法，在这个矩阵中，MATLAB 7.0 将不会存储矩阵中的 0 元素，而只对非 0 元素进行操作，这样就大大减少了计算机的存储空间和计算时间，下面将对稀疏矩阵进行介绍。

4.4.1 稀疏矩阵的生成

在 MATLAB 7.0 中，生成稀疏矩阵用特殊的函数来进行，这些函数有 speye、spones、spdiags、sparse、find、full、spalloc、sprand 和 sprandn 等，各函数的具体用法读者可以在 MATLAB 7.0 的帮助系统中查得。在运算中，MATLAB 7.0 将对稀疏矩阵采取不同于满矩阵的算法来进行计算。下面介绍几个常用的生成稀疏矩阵的函数。

◆ speye 函数

设 A 为单元矩阵，则 $speye(size(A))$ 生成和 A 维数相同的单位稀疏矩阵，而 $speye(M,N)$ 也生成一个单位稀疏矩阵，其维数为 M 和 N 中较小的那个数，$speye(M)$ 生成 M 阶的单位稀疏矩阵。

例 4-45　单位稀疏矩阵的生成。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> A=eye(10)
A =
     1    0    0    0    0    0    0    0    0    0
```

```
     0     1     0     0     0     0     0     0     0     0
     0     0     1     0     0     0     0     0     0     0
     0     0     0     1     0     0     0     0     0     0
     0     0     0     0     1     0     0     0     0     0
     0     0     0     0     0     1     0     0     0     0
     0     0     0     0     0     0     1     0     0     0
     0     0     0     0     0     0     0     1     0     0
     0     0     0     0     0     0     0     0     1     0
     0     0     0     0     0     0     0     0     0     1
>> speye(size(A))
ans =
   (1,1)        1
   (2,2)        1
   (3,3)        1
   (4,4)        1
   (5,5)        1
   (6,6)        1
   (7,7)        1
   (8,8)        1
   (9,9)        1
  (10,10)       1
>> speye(7,6)
ans =
   (1,1)        1
   (2,2)        1
   (3,3)        1
   (4,4)        1
   (5,5)        1
   (6,6)        1
>> speye(5)
ans =
   (1,1)        1
   (2,2)        1
   (3,3)        1
   (4,4)        1
   (5,5)        1
>>
```

- sprand 函数

该函数用于生成随机稀疏矩阵(其元素服从 0-1 分布)。

$R = sprand(S)$ 产生与稀疏矩阵 S 结构相同的稀疏矩阵 R，但是它的元素都是 0 到 1 上的随机数。

$R = sprand(M, N, D)$ 产生一个 $M \times N$ 的随机稀疏矩阵 R，它的非零元素的个数近似为 $M \times N \times D$，注意 D 的值在 0 到 1 之间且不要过大。

例 4-46　随机稀疏矩阵(0-1 分布)的生成。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> v=[3 5 6 2 1 9 6 5 5 6];
>> S=diag(v)
S =
     3     0     0     0     0     0     0     0     0     0
     0     5     0     0     0     0     0     0     0     0
     0     0     6     0     0     0     0     0     0     0
     0     0     0     2     0     0     0     0     0     0
     0     0     0     0     1     0     0     0     0     0
     0     0     0     0     0     9     0     0     0     0
     0     0     0     0     0     0     6     0     0     0
     0     0     0     0     0     0     0     5     0     0
     0     0     0     0     0     0     0     0     5     0
     0     0     0     0     0     0     0     0     0     6
>> R=sprand(S)
R =
   (1,1)        0.4418
   (2,2)        0.3533
   (3,3)        0.1536
   (4,4)        0.6756
   (5,5)        0.6992
   (6,6)        0.7275
   (7,7)        0.4784
   (8,8)        0.5548
   (9,9)        0.1210
  (10,10)       0.4508
>> R=sprand(10,10,0.08)
R =
   (3,1)        0.1708
   (3,2)        0.9943
  (10,2)        0.3932
   (8,3)        0.4398
   (9,3)        0.3400
   (9,7)        0.3142
   (9,9)        0.3651
>>
```

4.4.2　稀疏矩阵与满矩阵的相互转换

在 MATLAB 7.0 中，用来将稀疏矩阵和满矩阵相互转换的函数有 sparse、full 和 find 等 3 个函数，下面介绍这 3 种函数具体的使用方法。

◆ sparse 函数

$S = sparse(X)$ 将满矩阵或者稀疏 X 转化为稀疏矩阵 S。

$S = sparse(i, j, s, m, n, nzm)$ 生成 $m \times n$ 阶的稀疏矩阵 S，向量 s 的元素分布在以向量 i 的对应值和向量 j 的对应值为坐标的位置上。nzm 为矩阵存储的非零元的个数。

$S = sparse(i, j, s, m, n)$ 生成 $m \times n$ 的稀疏矩阵 S，向量 s 的元素分布在以向量 i 的对应值和向量 j 的对应值为坐标的位置上，其中 nzmax = length(s)。

$S = sparse(i, j, s)$ 生成 $m \times n$ 的稀疏矩阵 S，向量 s 的元素分布在以向量 i 的对应值和向量 j 的对应值为坐标的位置上，其中 $m = \max(i)$，$n = \max(j)$。

$S = sparse(m, n)$ 就是 $sparse([],[],[],m,n,0)$ 的简化形式。

例 4-47　用 sparse 函数将满矩阵转化为稀疏矩阵。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> i=[6 2 7 7 4 1 2 5];
>> j=[1 3 2 7 2 8 3 2];
>> s=[8 3 7 7 1 7 0 2];
>> X=diag(s,-2)
X =
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     8     0     0     0     0     0     0     0     0     0
     0     3     0     0     0     0     0     0     0     0
     0     0     7     0     0     0     0     0     0     0
     0     0     0     7     0     0     0     0     0     0
     0     0     0     0     1     0     0     0     0     0
     0     0     0     0     0     7     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     2     0     0
>> S=sparse(X)
S =
   (3,1)        8
   (4,2)        3
   (5,3)        7
   (6,4)        7
   (7,5)        1
   (8,6)        7
  (10,8)        2
>> S1=sparse(i,j,s,10,10,7)
??? Error using ==> sparse
Index exceeds matrix dimensions.
>> S1=sparse(i,j,s,10,10,8)              %注意 nzmax 的值的设定，如此处就不能是 7
S1 =
   (6,1)        8
   (4,2)        1
```

```
    (5,2)        2
    (7,2)        7
    (2,3)        3
    (7,7)        7
    (1,8)        7
>> S1=sparse(i,j,s,10,9)          %此处 nzmax = length(s)。
S1 =
    (6,1)        8
    (4,2)        1
    (5,2)        2
    (7,2)        7
    (2,3)        3
    (7,7)        7
    (1,8)        7
>> S1=sparse(i,j,s)               %此处 m = max(i) ， n = max(j)
S1 =
    (6,1)        8
    (4,2)        1
    (5,2)        2
    (7,2)        7
    (2,3)        3
    (7,7)        7
    (1,8)        7
>>
```

◆ full 函数

在 MATLAB 7.0 中，可以利用 full 函数将稀疏矩阵转化为满矩阵。

$S = full(X)$ 将稀疏矩阵 X 转化为满矩阵 S 。如果 X 本身是满矩阵，系统将不做任何操作。

例 4-48　full 函数的使用。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> s(6,1)=8;                  %稀疏矩阵的生成
   s(4,2)=1;
   s(5,3)=60;
   s(6,2)=57;
   s(2,3)=23;
   s(1,7)=25;
   s(3,8)=37;
>> full(s)                    %FULL 函数的使用
ans =

     0     0     0     0     0     0    25     0
     0     0    23     0     0     0     0     0
```

```
     0     0     0     0     0     0     0    37
     0     1     0     0     0     0     0     0
     0     0    60     0     0     0     0     0
     8    57     0     0     0     0     0     0
>>
```

◆ find 函数

$I = find(X)$ 返回矩阵 X 的非零元素的位置。例如，$I = find(X > 100)$ 返回 X 中大于 100 的元素的位置。

$[I, J] = find(X)$ 返回矩阵 X 中非零元素所在的行(I)和列(J)的具体数据。这种用法通常用于稀疏矩阵。

$[I, J, V] = find(X)$ 除了返回 I 和 J 以外还返回一个向量，这个向量中的数值为矩阵中非零元素数值。

注意：

$find(X)$ 和 $find(X \sim= 0)$ 会产生同样的 I 和 J，但是后者会生成一个包括所有非零元素位置的向量。

例 4-49　find 函数的应用。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>>  S(10,50)=82;
>>  S(32,14)=82;
>>  S(251,396)=25;
>> I=find(S)                %生成S中非零元素的位置
I =
        3295
       12309
       99396
>> [I,J] = find(S)          %生成S中非零元素的行列值
I =                         %I为行的值
    32
    10
   251
J =                         %J为列的值
    14
    50
   396
>> [I,J,V] = find(S)        %生成S中非零元素的行列值以及非零元素值
I =                         %I为行的值
    32
    10
   251
J =                         %J为列的值
```

```
      14
      50
     396
  V =                              %V 为非零元素值
      82
      82
      25
  >>
```

4.4.3 稀疏矩阵的操作

对稀疏矩阵进行操作，主要由 nnz、nonzeros、nzmax、sponse、spalloc、isspase、spyfun 和 spy 等函数来实现。下面对其中的几个函数进行详细介绍，读者可以依此类推，来学习其他函数的使用方法。

♦ nnz 函数

在 MATLAB 7.0 中，nnz 函数用于求非零元素的个数。

$nz = nnz(S)$ 返回矩阵 S 中非零元素的个数。

$D = nnz(S)/prod(size(S))$ 表示稀疏矩阵 S 中非零元素的密度。

例 4-50　nnz 函数的应用。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> v=[6 2 7 7 4 1 3 5];
>> S=diag(v,-1)
S =
     0     0     0     0     0     0     0     0     0
     6     0     0     0     0     0     0     0     0
     0     2     0     0     0     0     0     0     0
     0     0     7     0     0     0     0     0     0
     0     0     0     7     0     0     0     0     0
     0     0     0     0     4     0     0     0     0
     0     0     0     0     0     1     0     0     0
     0     0     0     0     0     0     3     0     0
     0     0     0     0     0     0     0     5     0
>> NZ=nnz(S)
NZ =
     8
>> D=nnz(S)/prod(size(S))
D =
    0.0988
>>
```

♦ sponse 函数

在 MATLAB 7.0 中，函数 sponse 用来将稀疏矩阵中的非零元素替换为 1。

$R = sponse(S)$ 生成一个与稀疏矩阵 S 结构相同的稀疏矩阵 R，但是在矩阵 S 中的非零元素的位置上用元素 1 替换。

例 4-51　sponse 函数的应用。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> S=sprandsym(10,0.05)
S =
    (9,3)        -1.6656
   (10,5)        -0.4326
    (8,7)         0.1253
    (7,8)         0.1253
    (3,9)        -1.6656
    (5,10)       -0.4326
>> R=spones(S)
R =
    (9,3)          1
   (10,5)          1
    (8,7)          1
    (7,8)          1
    (3,9)          1
    (5,10)         1
>>
```

♦ spalloc 函数

在 MATLAB 7.0 中，函数 spalloc 用来为稀疏矩阵分配空间。

$S = spalloc(m,n,nzm)$ 生成一个所有元素都为零的 $m \times n$ 阶稀疏矩阵，计算机利用这些空间来存储 nzm 的非零元素。

例如程序的说明如下。

```
s = spalloc(n,n,5*n);      %函数 SPALLOC 用来为稀疏矩阵分配空间
for j = 1:n                %变量 j 循环 n 次
s(:,j) = (含有 5 个非零元素的稀疏向量);
end
```

例 4-52　spalloc 函数的使用。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> n=3;
>> v=sprand(3,1,0.33)
v =
   (2,1)          0.0579
>> s = spalloc(n,n,1*n);
         for j = 1:n
```

```
                s( ,j) = (v)
        end
s =
   (2,1)        0.0579
s =
   (2,1)        0.0579
   (2,2)        0.0579
s =
   (2,1)        0.0579
   (2,2)        0.0579
   (2,3)        0.0579
>>
```

◆ issparse 函数

在 MATLAB 7.0 中，函数 issparse 用来判断矩阵是否为稀疏矩阵。

issparse(*S*) 返回值为 1 说明矩阵 *S* 是一个稀疏矩阵；返回值为 0 时说明矩阵 *S* 不为稀疏矩阵。

例 4-53　issparse 函数的使用。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> v=[6 2 7 8 8 1 2 4];
>> S=diag(v,2)                %生成矩阵 S
S =
     0     0     6     0     0     0     0     0     0     0
     0     0     0     2     0     0     0     0     0     0
     0     0     0     0     7     0     0     0     0     0
     0     0     0     0     0     8     0     0     0     0
     0     0     0     0     0     0     8     0     0     0
     0     0     0     0     0     0     0     1     0     0
     0     0     0     0     0     0     0     0     2     0
     0     0     0     0     0     0     0     0     0     4
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
>> R=sparse(S)                %生成稀疏矩阵 R
R =
   (1,3)        6
   (2,4)        2
   (3,5)        7
   (4,6)        8
   (5,7)        8
   (6,8)        1
   (7,9)        2
   (8,10)       4
>> N=issparse(S)
```

```
N =
       0                    %可知 S 不为稀疏矩阵
>> Y=issparse(R)           %可知 R 为稀疏矩阵
Y =
       1
>>
```

4.5 习　题

1. 令 $A=[1,2,3]$，$B=[3,1,4]$，$C=[9,-1,4]$。

(1) 求 A 和 B 的点积；

(2) 求 B 和 C 的叉积；

(3) 求 A、B 和 C 的混合积。

2. 设 $a=\begin{bmatrix}2 & -1\\ -2 & -2\end{bmatrix}$，$b=\begin{bmatrix}2 & -3\\ 0 & -4\end{bmatrix}$，$c=\begin{bmatrix}1\\ 2\end{bmatrix}$，$d=eye(2)$。

求解下列问题。

(1) $3\times a$

(2) $a+b$；

(3) $a*d$；

(4) $a.*d$；

(5) $a*c$；

(6) $a.*c$；

(7) $a\backslash b$；

(8) $a.\backslash b$；

(9) $a.\hat{}b$；

3. 求解 4 阶随机矩阵的特征值和特征向量。

4. 设 A 为 5 阶魔术矩阵，分别对 A 进行如下操作。

(1) 求 A 的逆；

(2) 求 A 的行列式；

(3) 求 A 的条件数；

(4) 求矩阵 A 的秩；

(5) 求矩阵 A 的迹。

5. X 为 4 阶随机矩阵，分别对 X 进行如下操作：

(1) 求 X 的 lu 分解；

(2) 求 X 的正交分解；

(3) 求 X 的特征值分解；

(4) 求 X 的 chol 分解；

(5) 求 X 的奇异值分解。

6. 设 A 为一个 4×3 的矩阵，试使用变维函数 reshape 将 A 变为一个 3×4 的矩阵。

7. B 为 5 阶范德蒙德矩阵，试使用矩阵变向函数实现 B 的 90 度旋转、上下翻转和左右翻转。

8. 利用表 4-2 提供的特殊函数表，生成 MATLAB 提供的各种特殊函数。

9. 首先生成一个 1×10 的随机数组 A，然后将它的第 5 个元素和第 8 个元素取出，并对它们进行四则运算。

10. 给习题 4.9 生成的数组分别进行升序和降序排序。

11. 比较稀疏矩阵与满矩阵的不同之处，如比较 eye(10)与 speye(10)生成矩阵的异同之处。

12. 将 10 阶随机矩阵转换为稀疏矩阵。

13. 将 10 阶稀疏正态随机矩阵转换为满矩阵。

第5章　单元数组和结构

在前面章节中已经介绍了几种 MATLAB 7.0 常用的数据类型，如数值型和字符型等。本章将介绍两种特殊的数据类型，即单元数组和结构，这两种数据类型的特点是允许用户将不同但是相关的数据类型集成到一个单一的变量。这样，因为相关的数据可以通过一个单元数组或是结构进行组织和访问，数据的管理就变得相对要容易一些。

5.1　单元数组

单元数组是 MATLAB 7.0 语言中比较特殊的一种数据结构，它的元素都以单元的形式存在。例如，可能它的某个单元的元素为实型，而另一个单元的元素为字符型。在程序中，每一个单元都是一个指向别的数据结构的指针。本小节将主要对单元数组的生成和运算做简要的介绍。

5.1.1　单元数组的生成

1. 直接生成单元数组

用类似矩阵的记号将给复杂的数据结构纳入一个变量之下。和矩阵中的圆括号表示下标类似，单元数组由大括号表示下标。

例 5-1　直接生成单元型变量。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> A={1,'Wind Gone',100+200*i,[90, 85, 55; 67, 70, 102, 57, 18, 100; -200, 89, 78]}
A =
      [1]      'Wind Gone'      [1.0000e+002 +2.0000e+002i]      [4x3 double]
```

2. 使用 cell 函数生成单元数组

在 MATLAB 7.0 中，可以用 cell 函数生成单元数组，具体的应用形式如下。

- $cell(N)$ 生成一个 $n \times n$ 阶的置空的单元数组；
- $cell(M,N)$ 或者 $cell([M,N])$ 生成一个 $m \times n$ 阶的置空的单元数组；
- $cell(M,N,P,...)$ 或者 $cell([M,N,P,...])$ 生成 $m \times n \times p...$ 阶的置空的单元数组；
- $cell(size((A))$ 生成与 A 同形式的单元型的置空矩阵。

例 5-2　使用 cell 函数生成单元数组。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> A=cell(2,2)
A
      []      []
```

```
        []        []
>> A{1,1}=[1 2 ;2 2]
A =
    [2x2 double]      []
         []           []
>> A{1,2}=['MATLAB 7.0']
A =
    [2x2 double]    'MATLAB 7.0'
         []              []
>> A{2,1}=['Tsinghua and Peking']
A =
              [2x2 double]    'MATLAB 7.0'
    'Tsinghua and Peking'             []
>> A{2,2}=[1+5*i,12-4*i]
A =
              [2x2 double]    'MATLAB 7.0'
    'Tsinghua and Peking'    [1x2 double]
>>
```

在本例中，首先使用 cell 函数定义了一个单元数组，此时，由于数组的各个元素还没有定义，因此，所显示的单元数组的元素就都用空的中括号来表示。然后，依次输入单元数组中各个元素的值，可以发现，每输入一个值，用相应的值代替空的中括号。

5.1.2　单元数组的操作

1. 单元数组的内容的显示或获取

欲对单元数组进行操作，首要的问题是要对单元数组的内容进行获取。只有这样才能进行更进一步的操作。

(1) elldisp 函数

在 MATLAB 7.0 中，celldisp 函数可以显示单元型变量的内容，具体应用形式如下。

- *celldisp*(*C*) 显示单元型变量 *C* 的内容；
- *celldisp*(*C*,'*name*') 在窗口中显示的单元型变量的内容的名称为 *name*，而不是通常的显示传统的 *ans*。

例 5-3　celldisp 函数的使用。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>>celldisp(A,'Huang')
Huang{1,1} =
     1     2
     2     2
Huang{2,1} =
Tsinghua and Peking
```

```
Huang{1,2} =
MATLAB 7.0
Huang{2,2} =
   1.0000 + 5.0000i   12.0000 - 4.0000i

>>
```

(2) cellplot 函数

在 MATLAB 7.0 中，cellplot 函数使用彩色的图形来显示单元型变量的结构形式，具体应用形式如下。

- *H = cellplot(C)* 返回一个向量，这个向量综合体现了表面、线和句柄；
- *H = cellplot(C,'legend')* 返回一个向量，这个向量综合体现了表面、线和句柄，并有图形注释 *legend* 。

例 5-4　cellplot 函数的使用。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> ou=cellplot(A)
ou =
   152.0016
   153.0011
   154.0011
   155.0011
   156.0011
   157.0011
>>
```

同时，MATLAB 7.0 将弹出图形窗口，生成显示单元型变量的结构形式的彩色图形，如图 5-1 示，读者可以观察图形各部分所代表的具体含义。

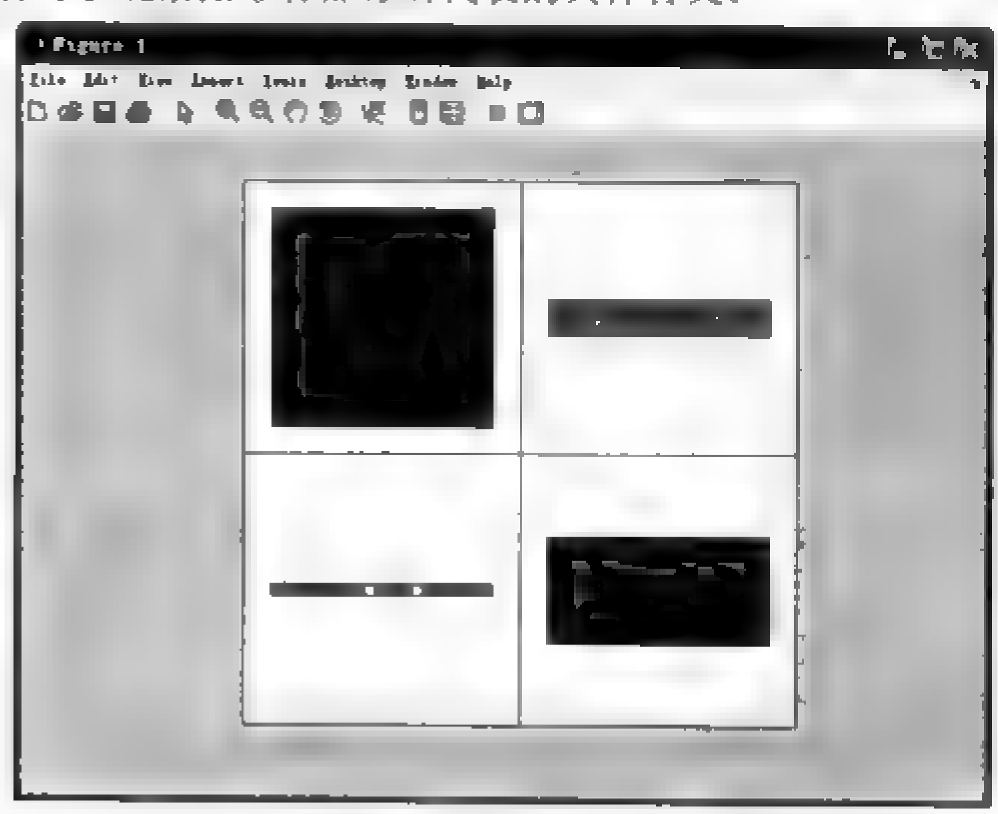


图 5-1　cellplot 函数生成的彩色图形

(3) 获取单元数组的内容

为了获取单元数组中一个单元的值，用户可以使用按值寻址，此时要用到大括号。应当注意的是，使用小括号与使用大括号是完全不同的，前者用于标识单元而不考虑这些单元的值，后者用于寻址单元的值。

例 5-5　使用花括号获取上例中单元数组的内容。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> A{1}
ans =
     1     2
     2     2
>> A{2}
ans =
Tsinghua and Peking
>> A{3}
ans =
MATLAB 7.0
>> A{4}
ans =
   1.0000 + 5.0000i   12.0000 - 4.0000i
>>
```

2. 单元数组的变维处理

♦ 添加或删除单元数组的单元

前边所述的对矩阵的变维处理同样也适用于对单元数组的变维处理。从某种意义上来说，单元数组的处理仅仅是应用于不同类型数组的处理技术的扩展。如果用户给当前单元数组的定义超出了其范围，MATLAB 7.0 将自动对这个数组进行扩展，并且用空的数值类型数组[ ]填充中间的其他新创建的数组单元。

使用中括号可以将多个单元数组集合成一个更大的数组，就像在数值型矩阵中中括号用来创建更大的数值型矩阵一样。

利用传统的数组寻址技术，可以将一个单元的子集提取出来生成一个新的单元数组。

使用空数组可以删除一个单元数组的整行或整列。

例 5-6　单元数组维数的扩展与缩小。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> A={[1 2 ;3 4],'love';100+i,[90, 85, 55; 67, 70, 102; 57, 18, 100; -200, 89, 78]}
A =
                            [2x2 double]        'love'
        [1.0000e+002 +1.0000e+000i]        [4x3 double]
>> B={[1 2 ;2 2],'MATLAB 7.0';'Tsinghua and Peking',[1+5*i,12-4*i]}
B =
                  [2x2 double]       'MATLAB 7.0'
```

```
        'Tsinghua and Peking'      [1x2 double]
>> C=[A,B]
C =
                    [2x2 double]     'love'                    [2x2 double]     'MATLAB
7.0'
        [1.0000e+002 +1.0000e+000i]     [4x3 double]     'Tsinghua and Peking'     [1x2 double]
>>
```

以上的程序段使用中括号将两个单元数组集成一个更大的数组；继续在命令窗口中输入如下程序，并按 Enter 键结束。

```
>> D=C(1,:)
D =
    [2x2 double]     'love'     [2x2 double]     'MATLAB 7.0'
>>
```

以上的程序段利用传统的数组寻址技术，将单元数组 C 的子集提取出来生成一个新的单元数组 D。继续在 MATLAB 7.0 中输入如下程序，并按 Enter 键结束。

```
>> C(2,:)=[]
C =
    [2x2 double]     'love'     [2x2 double]     'MATLAB 7.0'
>>
```

以上的程序段使用空数组删除单元数组 C 的第 2 行。

◆ reshape 函数改变单元数组的结构

函数 reshape 对单元数组的用法与对数值型矩阵的用法一样，它可以用于改变一个单元数组的结构，但不能用来添加或删除单元。

◇ $reshape(X,M,N)$ 命令将单元数组 X 的所有元素分配到一个 $M \times N$ 的新的单元数组，当单元数组 X 的元素不是 $M \times N$ 时，将返回一个错误。

◇ $reshape(X,M,N,P,...)$ 命令返回由单元数组 X 的元素组成的 $M \times N \times P \times ...$ 多维单元数组，如果 $M \times N \times P \times ...$ 与 X 的元素数不一样时，将返回错误。

◇ $reshape(X,[M,N,P,...])$ 命令与 $reshape(X,M,N,P,...)$ 命令的效果相同。

例 5-7　使用 reshape 函数改变上例中单元数组的结构。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> a=reshape(A,1,4)
a
    [2x2 double]     [1.0000e+002 +1.0000e+000i]     'love'     [4x3 double]
>> b=reshape(B,4,1)
b =
              [2x2 double]
```

```
        'Tsinghua and Peking'
        'MATLAB 7.0'
                  [1x2 double]
>>
```

在 MATLAB 7.0 中，与单元型变量相关的操作还有 iscell、cellfun 和 num2cell 等函数，读者可以查看 MATLAB 7.0 的帮助系统或是相关的资料做进一步了解。

5.2　结构型变量

结构型变量是另一种可以将不同类型数据组合在一起的数据类型。MATLAB 7.0 的结构体类似 C 语言的结构体数据结构。每个成员变量用指针操作符“.”表示，如 A.p 表示 A 变量的 p 成员变量。获得该成员比 C 更直观，仍用 A.p 访问，而不用 A->p。它的总的数据结构有一个名字，而且它的每一个成员元素都有自己的名字。

1. 结构型变量的生成

可以使用两种方法生成结构型变量，一种是在命令窗口中直接输入，还有一种是使用 struct 函数。

◆ 直接输入法

采用直接输入法时，在给结构体成员元素直接赋值的同时定义该元素的名称，并使用“.”将结构型变量名和成员元素名连接。

例 5-8　直接输入结构型变量。

下面的语句用来建立一个学生的小型数据库。

```
>> student.test=[99 56 96 87 67 69 87 76 92];
>> student.name='Huang Liang';
>> student.weight=67;
>> student.height=1.68;
>> student.num=034093;
>> student.add='School of civil engneering,Tsinghua university';
>> student.tel='13810498313';
```

输入 student，即可以在命令窗口中输出 student 的内容。

```
>> student
student =
      test: [99 56 96 87 67 69 87 76 92]
      name: 'Huang Liang'
      weight: 67
      height: 1.6800
      num: 34093
```

```
        add: 'School of civil engneering.Tsinghua university'
        tel: '13810498313'
>>
```

用户还可以通过以下形式的语句以添加新的结构型变量(另外的学生数据)。

```
>> student(2).test=[99 65 88 78 76 98 75 96 59];
%系统将默认之前的 student 为 student(1)
>> student(2).name='Wei Huan';
>> student(2).weight=50;
>> student(2).height=1.58;
>> student(2).num=034999;
>> student(2).add='School of Psychology.Chongqing university';
>> student(2).tel='02361701456';
```

此时，输入 student，将只得到该结构的成员变量名而不显示内容，

```
student
student =
1x2 struct array with fields:
    test
    name
    weight
    height
    num
    add
    tel
>>
```

可以继续在命令窗口中查询 student 的具体内容。

```
>> student(1)
ans =
        test: [99 56 96 87 67 69 87 76 92]
        name: 'Huang Liang'
        weight: 67
        height: 1.6800
        num: 34093
        add: [1x46 char]
        tel: '13810498313'
>> student(2)
ans =
        test: [99 65 88 78 76 98 75 96 59]
        name: 'Wei Huan'
        weight: 50
        height: 1.5800
```

```
        num: 34999
        add: 'School of Psychology.Chongqing university'
        tel: '02361701456'
>>
```

◆ 使用 struct 函数生成结构型变量

在 MATLAB 7.0 中，struct 函数可以产生一个结构体变量，最基本的使用方式是 $struct_array = struct('field', V1, 'field2', V2, ...)$，其中，$field$ 为各成员变量名，$V1$ 等为对应的各成员变量的内容。

例 5-9　使用 struct 函数生成结构型变量。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> struct_array=struct('countrys',{{'China','American'}},'strengths',[10000 9000])
struct_array =
     countrys: {'China'   'American'}
    strengths: [10000 9000]
>>
```

用户也可以一次输入多个变量的值，输入格式如下。

```
>> s = struct('type',{'big','little'},'color','red','x',{3 4})
s =
1x2 struct array with fields:
    type
    color
    x
```

2. 结构型变量的操作

◆ 在结构体变量中添加成员变量

如果用户想在一个结构体变量中添加一个新的成员变量，可以按照一定格式直接输入。例如，用户要在 student 中添加 gender 和 age 这两项记录，如例 5-10 所示。

例 5-10　在结构体变量中添加成员变量。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> student(1).gender='Male',         %在 student 中添加 gender 和 age 这 2 项记录
>> student(1).age=25;
>> student(2).gender='Female',
>> student(2).age=21;
>> student                            %查询 student 的结构
student
1x2 struct array with fields:
    test
    name
    weight
```

```
        height
        num
        add
        tel
        gender
        age
>>
```

由上面的输出可以看出，在 student 中新添加了 gender 和 age 这两项记录。

```
>> student(1)                    %输出 student(1)的具体内容
ans =
          test: [99 56 96 87 67 69 87 76 92]
          name: 'Huang Liang'
          weight: 67
          height: 1.6800
          num: 34093
          add: [1x46 char]
          tel: '13810498313'
          gender: 'Male'
          age: 25
>> student(2)                    %输出 student(2)的具体内容
ans =
          test: [99 65 88 78 76 98 75 96 59]
          name: 'Wei Huan'
          weight: 50
          height: 1.5800
          num: 34999
          add: 'School of Psychology.Chongqing university'
          tel: '02361701456'
          gender: 'Female'
          age: 21
>>
```

♦ 在结构体变量中删除成员变量

在 MATLAB 7.0 中，可以使用函数 rmfifld 从结构体变量中删除成员变量。函数的具体应用形式如下。

(1) $S = rmfifld(S,'field')$ 从 $m \times n$ 阶的结构体变量 S 中删除指定的成员变量 $field$。但该函数仍保留 S 原有的结构形式；

(2) $S = rmfifld(S, fields)$ 当 $fields$ 是字符型变量或单元型变量时，将一次性删除多个成员变量。

例 5-11 在结构体变量中删除成员变量。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> student=rmfield(student,'age')
student
1x2 struct array with fields:
     test
     name
     weight
     height
     num
     add
     tel
     gender
>> student(2)
ans =
        test: [99 65 88 78 76 98 75 96 59]
        name: 'Wei Huan'
        weight: 50
        height: 1.5800
        num: 34999
        add: 'School of Psychology,Chongqing university'
        tel: '02361701456'
        gender: 'Female'
>>
```

♦ 在结构体变量中调用成员变量

在 MATLAB 7.0 中，调用成员变量的方法十分简单。结构体变量中的任何信息，可以通过“结构体名称+成员变量名称”的格式进行取出；从结构体变量中取出的这些信息之后，可以再利用 MATLAB 7.0 中的相关函数进行调用。

例 5-12　在结构体变量中调用成员变量。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> student(1).test                    %从结构体变量中取出相关信息
ans =
     99    56    96    87    67    69    87    76    92
>> student(1).test(5)
ans =
     67
>> student(1).add
ans =
School of civil engneering,Tsinghua university
>> student(2).add
ans =
School of Psychology,Chongqing university
>> S=sum(student(1).test)             %对信息进行调用：求和与求平均
S =
```

```
    729
>> A=mean(student(1).test)
A =
     81
>>
```

需要注意的是，用户不能同时从多个结构体变量中取出某个成员变量。例如，表达式 student.name 将会导致错误。如果用户需要调用所有学生的名字，则必须使用循环语句 for(MATLAB 7.0 语言的使用在以后的章节将进行详细介绍)。

```
>> for j=1:length(student)
    disp(student(j).name);
    end
Huang Liang
Wei Huan
>> test_list=[],
>> for j=1:length(student)
    test_list=[test_list student(j).test]
    end
test_list =
     99    56    96    87    67    69    87    76    92
test_list =
   Columns 1 through 10
     99    56    96    87    67    69    87    76    92    99
   Columns 11 through 18
     65    88    78    76    98    75    96    59
>> mean(test_list)
ans =
    81.2778
```

◆ getefield 和 setfield 函数的使用

在 MATLAB 7.0 语言中，getefield 和 setfield 函数也是有关结构型变量的函数，具体的介绍如下。

getefield 函数取得当前存储在某个成员变量中的值。表达式 $F = getfield(S,'field')$ 返回指定成员变量的内容，与表达式 $F = S.field$ 等价，其中 S 必须为 1×1 阶的结构体变量，如上面所讨论的 student(1)和 student(2)就为 1×1 阶的结构体变量，而 student 则为 1×2 阶的结构体变量，表达式 $F = getfield(S,\{i,j\}'field',\{k\})$ 则等价于 $F = S(i,j).field(k)$。

setfield 函数给某个成员变量插入新的值。表达式 $S = setfield(S,'field',V)$ 将成员变量 $field$ 的值设置为 V 与表达式 $S.field = V$ 等价，其中 S 必须为 1×1 阶的结构体变量，表达式 $S = getfield(S,\{i,j\}'field',\{k\},V)$ 则等价于 $S(i,j).field(k) = V$。

例 5-13　getefield 和 setfield 函数的使用。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> NAMES=fieldnames(student)
%调出结构型变量 student 的所有成员变量名
NAMES =
    'test'
    'name'
    'weight'
    'height'
    'num'
    'add'
    'tel'
>> GETF=getfield(student(1),'add')
GETF =
School of civil engneering.Tsinghua university
>> SETF=setfield(student(2),'weight',80)
SETF =
      test: [99 65 88 78 76 98 75 96 59]
      name: 'Wei Huan'
      weight: 80
      height: 1.5800
      num: 34999
      add: 'School of Psychology.Chongqing university'
      tel: '02361701456'
>> YFIELD=isfield(student(1),'test') %判断是否为结构型变量的属性
YFIELD =
     1
>> YSTRU=isstruct(student(2))   %判断是否为结构型变量
YSTRU =
     1
>>
```

5.3 习　　题

1. 创建一个单元数组，使之包括两个实型变量，1 个字符型变量，1 个双精度变量。

2. 创建一个函数，使之可以接受任意类型的数值输入，并将各个单元的值相加，用户可以使用如下数值来检验该程序，$a=10$，$b=\begin{bmatrix}1\\12\\-50\end{bmatrix}$，$c=[10,-1,6]$。

(说明：用户可以在学习完本书的第 10 章之后再做本题。)

3. 创建一个结构型变量，用于对学生情况进行统计，包括学生的性别、年龄、民族、

入学成绩(包括数学、英语和专业 3 门功课)、身高和体重信息。然后使用该结构型变量对一个班级的学生进行数据管理。如计算入学成绩平均分、平均年龄、平均体重以及比较男女生的多少等。

第6章　字　符　串

字符和字符串是MATLAB语言的重要组成部分，MATLAB语言提供了强大的字符串处理功能，特别是在MATLAB 7.0中提供了专门的符号运算工具箱(Symbolic Math Toolbox)，使符号运算的功能更为强大。

6.1　设定字符串

在MATLAB 7.0中的字符串一般是ASCII值的数值数组，它作为字符串表达式进行显示。MATLAB 7.0对字符串的设定非常简单，只需用单引号(')将需设定的字符串引注即可。

例6-1　字符串的生成。

解：在命令窗口中输入如下命令，并按Enter键确认。

```
>> str='This is a string! It can be used easily!'
str =
This is a string! It can be used easily!
>> whos
  Name        Size                    Bytes  Class
  str         1x40                       80  char arra
Grand total is 40 elements using 80 bytes
>>
```

另外，字符串内的单引号是由两个连续的单引号来表示。字符串连接可以直接从数组连接中得到。继续在命令窗口中输入如下语句，并按Enter键确认。

```
 >> u='Taiwan is part of china,'
u =
Taiwan is part of china,
>> v='and Jiang core said:''There must be a war between the strait!'''
v =
and Jiang core said:'There must be a war between the strait!'
>> w=[u,v]
w =
Taiwan is part of china,and Jiang core said:'There must be a war between the strait!'
>>
```

如同矩阵，字符串可以有多个行，但每行必须有相同数目的列数。因此，要用空格以使所有行有相同长度，继续在命令窗口中输入如下语句，并按Enter键确认。

```
>> t=['China win 32 gold metals in the Olympic Games'
'and Chinese TAibei Team win 2 gold metals        '
'That doesn''t mean any thing                     ']
t
China win 32 gold metals in the Olympic Games
and Chinese TAibei Team win 2 gold metals
That doesn't mean any thing
 >>
```

如果各列输入的字符串长度不一致时，MATLAB 7.0 将给出错误的提示。继续在命令窗口中输入如下语句，并按 Enter 键确认。

```
>> t=['China win 32 gold metals in the Olympic Games'
      'and Chinese TAibei Team win 2 gold metals       '
      'That doesn''t mean any thing                    ']
??? Error using ==> vertcat
All rows in the bracketed expression must have the same
number of columns.
```

如果用户欲使各列的字符串长度不一致，可以使用 char 函数，具体情况可以参见例 6-5。

6.2 字符串的操作

6.2.1 字符串元素的读取

1. 利用数组操作工具进行读取

因为字符串是数值数组，可用 MATLAB 7.0 中所有可利用的数组操作工具进行读取。用户可以根据需要读取已经设定的字符串中的某一个元素或是多个元素。

例 6-2 读取上例中 str 中的元素。

解：首先读取 str 中的第 6 个元素。

在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> str(6)
ans =
i
>>
```

如果用户输入 str(5):

```
>> str(5)
ans =
```

```
>>
```

由上面的结果可以发现，在 MATLAB 7.0 中空格也是一个字符。可以一次读取字符串中的多个字符，继续在命令窗口中输入如下语句，并按 Enter 键确认。

```
>> s=str(10:16)
s =
 string
>> w=str(16:-1:10)
w =
gnirts                    %这是单词 string 的反向拼写。
>>
```

2. 使用 disp 函数显示字符串

函数 disp 允许不显示它的变量名而显示一个字符串。

例 6-3　使用 disp 函数显示字符串的内容，并与直接输出字符串内容做对比。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> disp(str)
This is a string! It can be used easily!
>> disp(s)
 string
>>
```

继续在命令窗口中输入如下语句，并按 Enter 键确认。

```
>> str
str =
This is a string! It can be used easily!
>> s
s =
 string
>>
```

由上边的程序段可以看出，使用 disp 函数显示字符串时，省去了“str =”和“s=”这两条语句，对脚本文件内显示帮助的文本是十分有用的。

6.2.2　字符串的基本变换

1. 字符串的 ASCII 码操作

一个字符串是由单引号括起来的简单文本。在字符串里的每个字符是数组中的一个元素，字符串的存储给每个字符分配 8 个字节，这是为了与其他的数据类型保持一致。因为 ASCII 字符只要求一个字节，故这种存储要求是浪费的，所分配的存储空间有 7/8 是无用

的。然而，对字符串保持同样的数据结构简化 MATLAB 7.0 的内部数据结构。所给出的字符串操作并不是 MATLAB 7.0 的基本特点，但这种表达是可接受的。

为了了解下面字符串的 ASCII 码表达，只需对字符串执行一些算术运算。最简单且计算上最有效的方法是取数组的绝对值。在 MATLAB 7.0 语言中，使用函数 abs 来求绝对值。

例 6-4 对例 6-1 中的字符串进行取绝对值运算。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> a=abs(str)
a =
  Columns 1 through 14
   84   104   105   115    32   105   115    32    97    32   115   116   114   105
  Columns 15 through 28
  110   103    33    32    73   116    32    99    97   110    32    98   101    32
  Columns 29 through 40
  117   115   101   100    32   101    97   115   105   108   121    33
>>
```

2. 使用 char 函数进行逆变换

MATLAB 7.0 的以前版本使用函数 setstr 来将 ASCII 码转换为字符串，在此版本中，setstr 函数仍然可以使用，但是在以后的版本中该函数可能会取消。MATLAB 7.0 同时提供了 char 函数来实现类似功能。下面介绍 char 函数的使用方法。对 setstr 函数的使用方法感兴趣的读者可以查看 MATLAB 7.0 的帮助系统或是查阅相关的文献。

- $S = char(X)$ 命令将包含正数(ASCII 码的前 127 个数码)的数组 X 转换为字符数组。
- 当 C 是一个字符型单元数组时，$S = char(C)$ 命令将 C 中的每一个单元转换为字符型数组的对应行。使用 cellstr 可以进行逆变换。
- $S = char(T1, T2, T3, \ldots)$ 命令生成的字符串矩阵包含字符串 $T1$、$T2$ 和 $T3$。它们的字符个数可以不相等。

例 6-5 使用 char 函数将 ASCII 码转换为字符串。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>>  t=['China win 32 gold metals in the Olympic Games'
'and Chinese TAibei Team win 2 gold metals       '
'That doesn''t mean any thing                    ']
t =
China win 32 gold metals in the Olympic Games
and Chinese TAibei Team win 2 gold metals
That doesn't mean any thing
>> s=abs(t)
s =
  Columns 1 through 14
```

```
    67   104   105   110    97    32   119   105   110    32    51    50    32   103
    97   110   100    32    67   104   105   110   101   115   101    32    84    65
    84   104    97   116    32   100   111   101   115   110    39   116    32   109
  Columns 15 through 28
   111   108   100    32   109   101   116    97   108   115    32   105   110    32
   105    98   101   105    32    84   101    97   109    32   119   105   110    32
   101    97   110    32    97   110   121    32   116   104   105   110   103    32
  Columns 29 through 42
   116   104   101    32    79   108   121   109   112   105    99    32    71    97
    50    32   103   111   108   100    32   109   101   116    97   108   115    32
    32    32    32    32    32    32    32    32    32    32    32    32    32    32
  Columns 43 through 45
   109   101   115
    32    32    32
    32    32    32
>> b=char(s)
b =
China win 32 gold metals in the Olympic Games
and Chinese TAibei Team win 2 gold metals
That doesn't mean any thing
>> s=char('China win 32 gold metals in the Olympic Games','and Chinese TAibei Team win 2 gold metals','That doesn''t mean any thing')
s =
China win 32 gold metals in the Olympic Games
and Chinese TAibei Team win 2 gold metals
That doesn't mean any thing
```

由于 MATLAB 7.0 语言是采用 C 语言开发的，因此它的字符串操作与 C 语言的相应操作基本相同。常见字符串的操作如表 6-1 所示。

表 6-1　常见字符串的操作

函数名	函数用途	函数名	函数用途
strcat	链接字符串	strvcat	垂直链接字符串
strcmp	比较字符串大小	strncmp	比较字符串的前 n 个字符
findstr	在其他的字符串中寻找该字符串	strjust	证明字符数组
strmatch	查找可能匹配的字符串	strrep	用其他字符串代替改串
strtok	查找字符串中的记号	blanks	生成空的字符串
deblank	删除字符串内的空格	ischar	字符串检验
iscellstr	字符串的单元检验	isletter	字母检验
isspace	空格检验	strings	strings 函数的帮助

3. 字符串的执行

在 MATLAB 7.0 中用 eval 函数来执行字符串。下面通过一个简单的实例来说明 eval 函数的使用方法。

例 6-6 用 eval 函数生成 3 到 5 阶的 magic 矩阵。

```
for n = 3 5
        eval(['M' num2str(n) ' = magic(n)'])
    end
```

在命令窗口中输入以上程序，并按 Enter 键确认，此时运算结果如下：

```
>> for n = 3:5
        eval(['M' num2str(n) ' = magic(n)'])
    end
M3 =
     8     1     6
     3     5     7
     4     9     2
M4 =
    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
M5 =
    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9
>>
```

6.2.3 字符串的运算

字符串的运算主要是指判断字符串是否相等，通过字符的运算来比较字符，字符串中字符的分类、查找与替换、字符串与数值的转换和数组与字符串的转换等。下面将分别给予详细的介绍。

1. 判断字符串是否相等

在 MATLAB 7.0 语言中，对两个字符串或字符串中的部分子字符串进行比较一般有下列 3 种情况。

- 比较两个字符串或字符串中的部分子字符串是否相等；
- 比较两个字符串中的个别字符是否相等；

- 先将字符串分成几个部分，判断每个部分是否为空白字符。

MATLAB 7.0 语言中，有两个函数可以用来判断两个输入的字符串是否相等。

- strcmp 函数：比较两个字符串是否相等，当相等时，系统将返回值 1，不相等时，返回值 0；
- strncmp 函数：比较两个输入字符串的前几个字符是否相等，当相等时，系统将返回值 1，不相等时，返回值 0。

例 6-7　判断字符串是否相等。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> words1='situate';
>> words2='situp';
>> N=strcmp(words1,words2)
N =
     0
>> Y=strncmp(words1,words2,3)
Y =
     1
>> N=strncmp(words1,words2,4)
N =
     1
>> N=strncmp(words1,words2,5)
N =
     0
>>
```

由上可知，字符串 words1 和字符串 words2 的内容显然不同，因此使用 strcmp 函数比较两个字符串是否相等时，返回值为 0；但是这两个字符串的前 4 个字符都相同，因此用 strncmp 函数比较两个输入字符串的前 4 个字符是否相等时，返回值为 1，而比较第 5 个字符是否相等时，返回值为 0。

```
>> famous1={'GREAT WALL';'SUMMER PALACE';'JIU ZHAI GOU'};
>> famous2={'GREAT WALL';'TAI MOUNTAIN';'ZHANG JIA JIE'};
>> N=strcmp(famous1,famous2)
N =
     1
     0
     0
>> N=strncmp(famous1,famous2,1)
N =
     1
     0
     0
>> N=strncmp(famous1,famous2,3)
```

```
N =
     1
     0
     0
>>
```

由上可知，当 strcmp 函数和 strncmp 函数用于单元数组时，函数会自动按照各单元的内容依次进行比较。

2. 通过字符的运算比较字符

当字符数组拥有相同的维数时，就可以利用 MATLAB 语言的运算法则，对字符数组进行比较。各运算符号的意义如表 6-2 所示。

表 6-2 运算符号的意义

符号形式	符号意义	英文简写
==	等于	eq
~=	不等于	ne
<	小于	lt
>	大于	gt
<=	小于或等于	le
>=	大于或等于	ge

例 6-8 通过字符的运算比较字符。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> F1='CARESS    ';
>> F2='CHARLATAN';
>> F1>=F2
ans =
     1     0     1     0     1     1     0     0     0
>>
```

由例 6-8 可知，用运算符号来比较字符数组是否相等时，将对两个字符数组中的字符逐个进行比较，运算符会根据字符所对应的 ASCⅡ码进行比较，当字符之间的关系满足运算符时，返回值为 1，反之为 0。

3. 字符串中字符的分类

在 MATLAB 7.0 语言中，字符串中的字符通常可以分为空白字符、字母字符和其他类型的字符。用户可以用两个函数来对字符串中的字符进行分类。

- *isspace*(*S*) 命令判断字符 *S* 是否为空白字符；
- *isletter*(*S*) 命令判断字符 *S* 是否为字母字符。

例 6-9　字符串中字符的分类。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> realstring='www03@mails.tsinghua.edu.cn www03@sohu.com'
realstring =
www03@mails.tsinghua.edu.cn www03@sohu.com
>> A=isspace(realstring)
A =
  Columns 1 through 11
     0     0     0     0     0     0     0     0     0     0     0
  Columns 12 through 22
     0     0     0     0     0     0     0     0     0     0     0
  Columns 23 through 33
     0     0     0     0     0     1     0     0     0     0     0
  Columns 34 through 42
     0     0     0     0     0     0     0     0     0
>> B=isletter(realstring)
B =
  Columns 1 through 11
     1     1     1     0     0     0     1     1     1     1     1
  Columns 12 through 22
     0     1     1     1     1     1     1     1     1     0     1
  Columns 23 through 33
     1     1     0     1     1     0     1     1     1     0     0
  Columns 34 through 42
     0     1     1     1     1     0     1     1     1
>>
```

可见，对于字符串中的空白字符，isspace (S)函数会返回 1；对于字母字符，isletter(S)函数会返回 1；而对于其他类型的字符，两者都会返回 0。

4. 查找与替换

查找与替换是字符串操作中的一项重要内容，MATLAB 7.0 语言提供了 findstr、strfind 和 strrep 等函数来实现查找与替换操作。

◆ findstr 函数

$K = findstr(S1, S2)$ 函数会根据所给的字符串中的字符来查找字符串，当查找成功后返回第一个相同字符的具体位置。$S1$ 和 $S2$ 的位置可以调换，即不管是 $S1$ 还是 $S2$ 都可以是被查找的对象。

例 6-10　findstr 函数的使用。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>>     s = 'How much wood would a woodchuck chuck?';
>> a1= findstr(s,'a')
```

```
a1 =
     21
>> a2= findstr(s,'a')
a2 =
     21
>> a3 =findstr(s,'a')
a3 =
     21
>> a4=findstr(s,'wood')
a4 =
     10     23
>> a5= findstr(s,'Wood')
a5 =
      []
>> a6=findstr(s,' ')
a6 =
      4      9     14     20     22     32
>>
```

◆ strfind 函数

$K = strfind(text, pattern)$ 函数会根据所给的字符串中的字符来查找字符串，当查找成功后会返回第一个相同字符的具体位置。但是 $S1$ 和 $S2$ 的位置不可以调换， 只能在字符串 *text* 中查找字符串 *pattern*。当 *pattern* 的长度大于 *text* 时会返回“[ ]”。

例 6-11 strfind 函数的使用。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> s = 'How much wood would a woodchuck chuck?';
>> b1= strfind(s,'a')
b1 =
     21

>> b2=strfind('a',s)
b2 =
      []
>> b3=strfind(s,'wood')
b3 =
     10     23
>> b4=strfind(s,'Wood')
b4 =
      []
>> b5= strfind(s,' ')
b5 =
      4      9     14     20     22     32
>>
```

◆ strrep 函数

$S = strrep(S1,S2,S3)$ 函数会把字符串 $S1$ 中的 $S2$ 子串都都换成字符串 $S3$，并返回置换后的新字符串。

当 $S1$、$S2$ 和 $S3$ 都是是单元型变量时，$strrep(S1,S2,S3)$ 返回一个与 $S1$、$S2$ 和 $S3$ 相同型号的单元型变量。此时要保证 $S1$、$S2$ 和 $S3$ 的型号相等。

例 6-12　strrep 函数的使用。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> s1='This is a beautiful women!'
s1 =
This is a beautiful women!
>> strrep(s1,'women','girl')
ans =
This is a beautiful girl!
>> strrep(s1,'beautiful','ugly')
ans =
This is a ugly women!
>> strrep(s1,'','women')
ans =
This is a beautiful women!
>>
```

5. 字符串与数值的转换

在 MATLAB 7.0 语言中，可以使用 num2str、int2str、str2num 和 str2double 等函数实现字符串和数值之间的相互转换。此外，还可以使用 hex2num 和 hex2dec 等函数实现十进制、十六进制和二进制数字之间的转换，如表 6-3 所示。

表 6-3　常见的字符串转换函数

函　数	功　能	函　数	功　能
hex2dec	将 16 进制字符串转化为 10 进制整数	dec2hex	将 10 进制整数转化为 16 进制字符串
bin2dec	将 2 进制字符串转化为 10 进制整数	dec2bin	将 10 进制整数转化为 2 进制字符串
base2dec	转化 B 底字符串为 10 进制整数	hex2num	将 16 进制字符转化为双精度数
upper	改该字符串为大写	lower	改该字符串为小写
fprintf	把格式化的文本写到文件中或显示屏上	sprintf	用格式控制，数字转换成字符串
sscanf	用格式控制，字符串转换成数字	char	ASCII 码转换成字符串
num2str	数字转换成字符串	int2str	整数转换成字符串

下面对其中一些函数的使用方法进行介绍。

◆ num2str 函数

num2str 函数用于将数字转换成字符。$T = num2str(X)$ 可以将矩阵 X 转换为一个字符串 T，其中 T 的精度为 4 位小数。$T = num2str(X, N)$ 将矩阵 X 转换为一个字符串 T，其中 T 的精度为 N 为小数；$T = num2str(X, format)$ 转换为 $format$ 格式的字符串。

例 6-13 num2str 函数的使用。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> X= num2str(randn(6,6))
X =
   -1.1465    -0.18671     1.0668     0.71432      -1.441      1.2902
    1.1909     0.72579   0.059281      1.6236     0.57115      0.6686
    1.1892    -0.58832  -0.095648    -0.69178    -0.39989      1.1908
-0.037633      2.1832   -0.83235       0.858        0.69     -1.2025
   0.32729     -0.1364    0.29441       1.254     0.81562    -0.01979
   0.17464     0.11393    -1.3362     -1.5937     0.71191    -0.15672
>> >> num2str(pi,10)
ans =
3.141592654
>> num2str(pi,30)                          %将圆周率输出位数设定为 30 位
ans =
3 1415926535897931                         %可见最多的输出位数为小数点后 16 位
>>
```

◆ int2str 函数

int2str 函数用于将指数转换为字符串，$S = \text{int}\,2str(X)$ 先将矩阵 X 中的元素取整之后再将其转换为一个字符串矩阵 S。

例 6-14 int2str 函数的使用。

```
>> X= int2str(randn(6,6))
X =
-2   0  1  0  0  1
 0  -1  2  0  1  0
-1  -2  1  0  1  0
 1   0 -1  0  1  0
-1  -1  0  1  0 -1
 1  -1 -2  1  2  2
>>
```

◆ str2num 函数

str2num 函数用于将字符串矩阵转换为数字矩阵。$X = \text{int}\,2str(S)$ 将一个字符型矩阵 S 转换为一个数字矩阵。

例 6-15 使用 str2num 函数将字符型矩阵转换为数字矩阵。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> S = ['1 2'   ;'3 4']
S =                                  % S 为字符型矩阵
1 2
3 4
>> X =str2num(S)
X =                                  %X 为数字型矩阵
      1      2
      3      4
>>
```

字符型矩阵 S 中的元素必须为 ASCII 码中的数字字符。每一个元素可以包括数字、小数点以及数字前的“+”或“-”，也可以是指数形式的字符或复数型字符。

如果字符串矩阵 S 中含有不符合上述规定的元素，$str2num(S)$ 将返回一个空矩阵，而 $[X,OK]=str2num(S)$ 将返回 $OK=0$。

需要注意的是，str2num 使用 eval 来转换输入语句，所以当字符串含有子函数时将产生边际效应，可以使用 str2double 函数来避免边际效应的产生。

同时空格的作用也是很重要的，例如：输入 str2num('1+2i')和 str2num('1 + 2i') 命令会得出 $x = 1+2i$，输入 str2num('1 +2i') 命令会得出 $x = [1\ 2i]$，而当用户使用 str2double 时可以避免类似问题的产生。

例 6-16　使用 str2num 函数将字符型矩阵转换为数字矩阵的另外几种形式。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> S = ['s e'   ;'3 4']
S =
s e
3 4
>> X=str2num(S)
X =
      []
>> [X,OK]=str2num(S)
X =
[]
OK =
0
>> str2num('1+2i')        %由下边的几行程序可以看出，空格的使用对输出结果有很大的影响
ans =
   1.0000 + 2.0000i
>> str2num('1 + 2i')
ans =
   1.0000 + 2.0000i
>> str2num('1 +2i')
```

```
ans =
    1.0000                    0 + 2.0000i
>>
```

♦ str2double 函数

str2double 函数用于将字符串转换为双精度的数值。*str2double(S)* 命令用于将字符串矩阵转换为数字矩阵，字符型矩阵 S 中的元素必须为 ASCII 码中的数字字符。每一个元素可以包括数字、小数点以及数字前的“+”或“ ”，也可以是指数形式的字符或复数型字符。

如果字符串矩阵 S 中含有不符合上述规定的元素，*str2num(S)* 将返回 NaN 。

$X = str2double(C)$ 将单元型字符矩阵 C 中的元素转换为双精度数值矩阵 X ，X 的型号与 C 一样。

例 6-17 str2double 函数的使用。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> str2double('123.45e7')
ans =
   1.2345e+009
>> str2double('123 + 45i')
ans =
   1.2300e+002 +4.5000e+001i
>> str2double('3.14159')
ans =
      3.1416
>> str2double('2.7i - 3.14')
ans =
  -3.1400 + 2.7000i
>> str2double({'2.71' '3.1415'})
ans =
     2.7100      3.1415
>> str2double('1,200.34')
ans =
   1.2003e+003
>>
```

6. 数组与字符串的转换

在 MATLAB 7.0 语言中用户可以使用 mat2str 函数将数组转换为字符串。mat2str 可以将一个二维的矩阵转换为相应的字符串。*STR = mat2str(MAT)* 命令将二维矩阵 *MAT* 转换为一个字符串 *STR* 。这样 *eval(STR)* 将返回具有 15 位精度的初始矩阵。当 *MAT* 中含有为非数量的元素时，矩阵 *MAT* 将被转换为“[]”。

例 6-18 mat2str 函数的使用。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> MAT=magic(6)
MAT =
    35     1     6    26    19    24
     3    32     7    21    23    25
    31     9     2    22    27    20
     8    28    33    17    10    15
    30     5    34    12    14    16
     4    36    29    13    18    11
>> STR=mat2str(MAT)
STR =
[35 1 6 26 19 24;3 32 7 21 23 25;31 9 2 22 27 20;8 28 33 17 10 15;30 5 34 12 14 16;4 36 29 13 18 11]
>>
```

6.3 习　　题

1. 编制一个程序，该程序将接受用户输入的任何字符串，并查找在这个字符串中某个指定的字符出现的次数。

2. 修改习题 6.1 中所编制的程序，使得在查找字符时，不区分字母的大小写。

3. 编制一个程序，使得该程序接受用户输入的字符串，并将该字符串按升序排列并打印出来。

4. MATLAB 7.0 提供了 upper 函数和 lower 函数，可以实现字母大小写的转换，并编制一个程序 caps，该程序可以将每个单词的第一个字母变为大写，其余的字母变为小写。

第7章　多　项　式

MATLAB 7.0 语言提供了一些处理多项式的专用函数，用户可以使用它们很方便地求解多项式的根，并且可以很容易地对多项式进行四则运算、积分和微分运算。

7.1　多项式的创建

在 MATLAB 7.0 语言中，对于多项式 $P=a_0x^n+a_1x^{n-1}+a_2x^{n-2}+\cdots+a_{n-1}x+a_n$，约定可以用向量 $P=[a_0,a_1,a_2\cdots a_{n-1},a_n]$ 来表示，这样，多项式问题就被转换为向量问题。

在 MATLAB 7.0 中，可以使用 3 种方法创建多项式，即直接输入系数向量法、特征多项式输入法和由根创建法。下面对它们分别予以介绍。

7.1.1　直接输入系数向量创建多项式

由于在 MATLAB 7.0 中多项式是以向量的形式存储的，因此，直接输入向量，MATLAB 7.0 将按降幂自动把向量的元素分配给多项式各项的系数。而该向量可以是行向量，也可以是列向量。

例 7-1　使用直接输入法创建多项式 $3x^5+5x^4+x^2+12$。

解：首先创建系数向量，注意缺少的各项在向量中以 0 代替。然后使用 poly2sym 函数将该向量转换为多项式，使用 disp 函数不显示“y=”。

```
>> P=[3 5 0 1 0 12]
P =
     3     5     0     1     0    12
>> y=poly2sym(P)
y =
3*x^5+5*x^4+x^2+12
>> disp(y)
3*x^5+5*x^4+x^2+12
>>
```

7.1.2　特征多项式输入法

MATLAB 7.0 提供了 poly 函数，使用它可以由矩阵的特征多项式创建多项式。使用该方法生成多项式时，其首项的系数必为 1。其使用格式如下。

- 当 A 是一个 N×N 矩阵时，*poly*(*A*) 命令求出 A 的特征多项式 det(*lambda*×*eye*(*size*(*A*)) − *A*)。
- 若 V 是向量，命令 *poly*(*A*) 生成以 V 为根的多项式。

例 7-2　使用 poly 函数求矩阵特征值对应的多项式。

解：在命令窗口中输入如下程序，并按 Enter 键确认。

```
>> A=[3 1 4 1; 5 9 2 6;5 3 5 8; 9 7 9 3]
A =
     3     1     4     1
     5     9     2     6
     5     3     5     8
     9     7     9     3
>> p=poly(A)
p =
    1.0000  -20.0000  -16.0000  480.0000    98.0000
>> disp(poly2sym(p))
x^4-20*x^3-16*x^2+480*x+1724034232352773/17592186044416
>>
```

7.1.3　由多项式的根逆推多项式

如果已知某个多项式的根，那么，使用 poly 函数，可以很轻松地产生其对应的多项式。其使用方法与上例相同。

例 7-3　根据给定的多项式的根逆推对应的多项式。

解：在命令窗口中输入如下程序，并按 Enter 键确认。

```
>> roots=[-4 -2+2i -2-2i 5]
roots =
  -4.0000             -2.0000 + 2.0000i  -2.0000 - 2.0000i    5.0000
>> p=poly(roots)
p =
     1     3   -16   -88  -160
>> disp(poly2sym(p))
x^4+3*x^3-16*x^2-88*x-160
>>
```

7.2　多项式的运算

对多项式的运算主要包括求多项式的值、四则运算和求根等。下面将对它们进行介绍。

7.2.1 多项式的求值

MATLAB 7.0 提供了两个函数来对多项式进行求值，即 polyval 和 polyvalm。前者以数组为计算单位，后者以矩阵为计算单位。下面对它们的使用方法予以介绍。

- $Y = polyval(P, A)$ 命令计算以向量 P 为系数的多项式在点 X 的值。$Y = P(1)\times X^n + P(2)\times X^{n-1} + ... + P(N)\times X + P(N+1)$，如果 X 是矩阵或是向量，那么该命令将对矩阵或是向量的每一个元素都进行计算。
- $Y = polyval(P, X, [], MU)$ 命令使用 $xhat = (X - MU(1))/MU(2)$ 来代替 X。其中 MU 是由 polyfit 函数算出来的随机结果。
- $Y = polyvalm(P, X)$ 命令计算以 P 为系数的多项式在方阵 X 的值。X 必须为方阵：$Y = P(1)\times X^n + P(2)\times X^{n-1} + ... + P(N)\times X + P(N+1)\times I$。

例 7-4　使用 polyval 和 polyvalm 函数求多项式的值。

解：首先是定义多项式的系数和求值点。

```
>> p=[1.0000   -20.0000   -16.0000   480.0000    98.0000]
p =
     1    -20    -16    480     98
>> x=4
x =
    4
>> polyval(p,x)                    %使用 polyval 函数求 x 点处的多项式值
ans =
    738
>>
```

为了比较两个函数的区别，这里再定义一个矩阵，分别使用两个函数进行求解，用户可以观察使用它们分别进行求解的区别。

```
>> X=magic(3)
X =
     8     1     6
     3     5     7
     4     9     2
>> polyval(p,X)
ans =
        -3230          543         -622
          935          223        -1785
          738        -4897          850
>> polyvalm(p,X)
ans =
        -4199        -4489        -4489
        -4489        -4199        -4489
```

```
        -4489        -4489        -4199
>>
```

7.2.2 求多项式的根

求多项式的根，即多项式为零的值，可能是许多学科共同的问题。MATLAB 7.0 求解这个问题，并提供其他的多项式操作工具。在 MATLAB 7.0 语言里，多项式由一个行向量表示，设为 p，它的系数按降序排列，使用 roots 函数可以求出该多项式的根。其使用格式为 roots(p)。

例 7-5　使用 roots 函数求多项式 $x^4+3x^2+12x-7$ 的根。

解：在命令窗口中输入如下程序，并按 Enter 键确认。

```
>> p=[1 0 3 12 -7]
p =
     1     0     3    12    -7
>> roots(p)
ans =
   0.7876 + 2.4351i
   0.7876 - 2.4351i
  -2.0872
   0.5121
>>
```

7.2.3 多项式的四则运算

1. 加法和减法

对多项式加法和减法，MATLAB 7.0 不提供专用的函数。如果两个多项式的向量阶数相同，标准的数组加法有效。当两个多项式的向量阶数不同时，需要在低阶多项式的前边补 0，使得它与相加的高阶多项式有相同的阶数。

例 7-6　多项式的加法和减法。

解：在命令窗口中输入如下程序，并按 Enter 键确认。

```
>> a=[8 2 2 8],b=[6 1 6 1]
a =
     8     2     2     8
b =
     6     1     6     1

>> c=a+b
c =
    14     3     8     9
```

```
>> Y1=poly2sym(a)
Y1 =
8*x^3+2*x^2+2*x+8
>> Y2=poly2sym(b)
Y2 =
6*x^3+x^2+6*x+1
>> Y3=poly2sym(c)
Y3 =
14*x^3+3*x^2+8*x+9
>>
```

由于向量 a 和 b 阶数相同，所以以上的程序实现了将它们直接相加。当几个相加的多项式的向量阶数不同时，需要在低阶多项式的前边补 0，使得它与相加的高阶多项式有相同的阶数。继续在命令窗口中输入如下程序，并按 Enter 键确认。

```
>> d=[2 4 5]
d =
     2     4     5
>> c+d
??? Error using ==> plus                    %因为 c 和 d 阶数不相等，所以提示错误
Matrix dimensions must agree.
>> c+[0 [d]]
ans =
    14     5    12    14
>> poly2sym(ans)
ans =
14*x^3+5*x^2+12*x+14
>>
```

2. 乘法

在 MATLAB 7.0 语言中，使用 conv 函数对多项式进行乘法运算。其使用格式为 $c = conv(a,b)$，其中 a 和 b 为两个多项式的系数向量，c 为相乘所生成的多项式的系数向量。

例 7-7 使用 conv 函数求多项式的积。

解：在命令窗口中输入如下程序，并按 Enter 键确认。

```
>> a=[1 2 3 4],b=[5 6 7 8]
a =
     1     2     3     4
b =
     5     6     7     8
>> Y1=poly2sym(a)
Y1 =
x^3+2*x^2+3*x+4
```

```
>> Y2=poly2sym(b)
Y2 =
5*x^3+6*x^2+7*x+8
>> c=conv(a,b)
c
      5     16     34     60     61     52     32
>> Y=poly2sym(c)
Y =
5*x^6+16*x^5+34*x^4+60*x^3+61*x^2+52*x+32
>>
```

当要实现两个以上的多项式的乘法时，用户可以重复使用 conv 函数进行运算。

3. 除法

在数值计算中，经常需要用一个多项式去除另一个多项式。在 MATLAB 7.0 语言中，使用 decon 函数来完成该项功能。

例 7-8　使用上例中的多项式实现多项式的除法运算。

解：在命令窗口中输入如下程序，并按 Enter 键确认。

```
>> d=deconv(c,a)
d =
      5      6      7      8
>> e=deconv(c,b)
e =
    1.0000    2.0000    3.0000    4.0000
>> x=poly2sym(d),y=poly2sym(e)
x =
5*x^3+6*x^2+7*x+8
y =
x^3+2*x^2+3*x+4
>>
```

本例实现的运算其实是例 7-7 的逆运算。

4. 求导和积分

在 MATLAB 7.0 语言中，分别使用 polyder 函数和 polyint 函数来求多项式的导数与积分。其使用格式如下。

- 若 P 为多项式的系数向量，$polyder(P)$ 命令对该多项式求导并返回求导后的系数向量。
- $polyder(A,B)$ 命令相当于 $polder(A*B)$ 命令。
- $[Q,D]=polyder(B,A)$ 命令返回多项式 B/A 的求导值，并以 Q/D 的形式表示出来。
- $poly\,\mathrm{int}(P,K)$ 命令返回多项式 P 的积分，以 K 为积分步。

♦ $poly\text{int}(P)$ 命令返回多项式 P 的积分，以 $K=0$ 为积分步。

例 7-9 分别使用 polyder 函数和 polyint 函数求多项式的导数和积分。

解：在命令窗口中输入如下程序，并按 Enter 键确认。

```
>> p=[3 1 8 8],a=[9 4 9 4],b=[8 4 6 7]
p =
     3     1     8     8
a =
     9     4     9     4
b =
     8     4     6     7
>> q=polyder(p)
q =
     9     2     8
>> p1=polyint(q)
p1 =
     3     1     8     0
>> w=polyder(a,b)
w =
   432   340   568   465   196    87
>> t=polyint(q,1)
t =
     3     1     8     1
>>
```

7.3 习　　题

1. 使用 7.1 节提供的 3 种方法创建多项式 $8X^6+2X^5+2X^4+6X^3+X^2+X$。

2. 使用两种方法求解方程 $X^4+9X^3+8X^2+1=0$ 的所有根。

3. 求习题 7.1 所提供的多项式在 $X=1$、$X=\pm 2$ 和 $X=\pm 12$ 处的值。

4. 设两个多项式的系数分别为 $a=[5,6,8,2]$，$b=[1,7,3,2]$，对这两个多项式进行如下操作。

(1) $a+b$

(2) $a\times b$

(3) a/b

(4) 求 a 的积分和 b 的导数。

第8章　关系和逻辑运算

除了传统的数值运算，MATLAB 7.0 语言还支持关系和逻辑运算。MATLAB 7.0 提供了一些关于逻辑运算的操作符和函数，这些操作符和函数用于提供求解真假命题的答案。逻辑运算的一个重要的应用在于控制基于真假命题的一系列命令(通常在 M 文件中)的流程，或决定执行次序。

作为所有关系和逻辑表达式的输入，MATLAB 7.0 把任何非零数值当作真，而只把零当作假。所有关系和逻辑表达式的输出，当结果为真时输出为 1；当结果为假时输出为 0。

8.1　关系操作符

MATLAB 7.0 语言的关系操作符能用来比较两个同样大小的数组，或用来比较一个数组和一个标量。在后一种情况，标量和数组中的每一个元素相比较，结果与数组大小一样。MATLAB 7.0 语言的关系操作符包括所有常用的比较运算符，如表 8-1 所示。

表 8-1　关系操作符

关系运算符	该运算符的功能	关系运算符	该运算符的功能
<	小于	>=	大于等于
>	大于	==	等于
<=	小于等于	~=	约等于

例 8-1. 关系操作符的运用。

解：下面给出几个示例。

在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> A=linspace(1,10,10)
A =
     1     2     3     4     5     6     7     8     9    10
>> B=linspace(10,1,10)
B =
    10     9     8     7     6     5     4     3     2     1
>> lj=A>B              %比较 A 和 B
lj =
     0     0     0     0     0     1     1     1     1     1
>> lj1=A>=6            %比较 A 和 6
```

```
lj1 =
     0     0     0     0     0     1     1     1     1     1
>> lj2=A==B              %比较 A 和 B 是否有相等的元素
lj2 =
     0     0     0     0     0     0     0     0     0     0
>>
```

上边的程序中最后一项是找出 A 中的元素等于 B 中的元素。用户应当注意的是，“=”和“==”在 MATLAB 语言中具有完全不同的意义：“= ”用来将运算的结果赋给一个变量；“== ”用来比较两个变量，当它们相等时返回 1，当它们不相等时返回 0。

8.2　逻辑操作符

逻辑操作符的功能在于使用“与”或者是“或”的功能将多个表达式组合在一起，或是是对关系表达式取反。MATLAB 7.0 语言提供的逻辑操作符包括以下 3 个，如表 8-2 所示。

表 8-2　逻辑操作符及其功能

逻辑操作符	功　能
&	与
|	或
~	非

下面介绍一些关于逻辑操作符用法的例子。

例 8-2　逻辑操作符的用法。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> A=linspace(1,10,10)
A =
     1     2     3     4     5     6     7     8     9    10
>> B=linspace(10,1,10)
B =
    10     9     8     7     6     5     4     3     2     1
>>
```

以上的程序生成了两个向量 A 和 B，下面利用生成的两个向量进行逻辑运算，继续在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> L1=A>4&B<7
L1 =
     0     0     0     0     1     1     1     1     1     1
>> L2=A>5&B>5
```

```
L2 =
     0     0     0     0     0     0     0     0     0     0
>> L3=A>5|B>5
L3 =
     1     1     1     1     1     1     1     1     1     1
>> L4=~A>6
L4 =
     0     0     0     0     0     0     0     0     0     0
>>
```

8.3　关系与逻辑函数

前面两小节介绍了 MATLAB 语言的关系与逻辑操作符，除了这些用户经常使用的符号外，MATLAB 7.0 语言还提供了大量的其他关系与逻辑函数，如表 8-3 所示。

表 8-3　关系与逻辑函数及其功能

关系和逻辑函数	使 用 功 能
xor(s,t)	异或运算，s 或 t 非零(真)返回 1，s 和 t 都是零(假)或都是非零(真)返回 0
any(x)	如果在一个向量 x 中，任何元素是非零，返回 1；矩阵 x 中的每一列有非零元素，返回 1
all(x)	如果在一个向量 x 中，所有元素非零，返回 1；矩阵 x 中的每一列所有元素非零，返回 1

下面对这些函数的使用举例说明。

例 8-3　xor、any 和 all 3 个函数的应用。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> s=[1 0 -1 3 5],
>> s=[2 1 0 4 0],
>> t=[1 0 -1 3 0],
>> x=[1 2 3 4];
>> y=[0 1 -1 0];
>> xor(s,t)
ans =
     0     1     1     0     0
>> any(x)
ans
     1
>> any(y)
ans =
     1
```

```
>> all(x)
ans =
     1
>> all(y)
ans =
     0
>>
```

此外，MATLAB 7.0 语言还提供了大量的测试函数，测试特殊值或条件的存在，并返回逻辑值，如表 8-4 所示。

表 8-4　测试函数及其功能

测试函数	函数功能
isfinite	元素有限时，返回真值；否则，返回假值
isempty	矩阵为空时，返回真值；否则，返回假值
isglobal	元数为全局变量时，返回真值；否则，返回假值
ishold	当前绘图保持状态是 'ON "时，返回真值；否则，返回假值
isieee	计算机执行 IEEE 算术运算时，返回真值；否则，返回假值
isinf	元素无穷大时，返回真值；否则，返回假值
isletter	元素为字母时，返回真值；否则，返回假值
isnan	元素为不定值时，返回真值；否则，返回假值
isreal	元素无虚部时，返回真值；否则，返回假值
isspace	元素为空格字符时，返回真值；否则，返回假值
isstr	元素为一个字符串时，返回真值；否则，返回假值
ismember	元素为某个集合中的元素时，返回真值；否则，返回假值
isnumeric	元素为数值型数组时，返回真值；否则，返回假值
isprime	元素为质数时，返回真值；否则，返回假值
issparse	矩阵为系数矩阵时，返回真值；否则，返回假值
iskeyword	元素为关键词或是保留字时，返回真值；否则，返回假值
ishandle	如果是图形句柄，返回真值；否则，返回假值
iscell	如果是单元数组，返回真值；否则，返回假值
isfield	如果是结构数组中的域，返回真值；否则，返回假值
isobject	如果是对象，返回真值；否则，返回假值
isstruct	如果是结构，返回真值；否则，返回假值

8.4　NaNs 和空矩阵

NaNs 和空矩阵([　])要求在 MATLAB 7.0 中作特殊处理，特别是用在逻辑或关系表达

式里，根据 IEEE 数学标准。对 NaNs 的几乎所有运算结果都得出 NaNs。

8.4.1　NaNs 的处理

在逻辑或关系表达式里，对 NaNs 的几乎所有运算结果都得出 NaNs。本节将通过实例讲述在 MATLAB 7.0 中怎样对 NaNs 进行操作。

例 8-4　对 NaNs 的逻辑或关系处理。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> x=[10   2   nan   inf  ]
x =
     10      2    NaN    Inf
>> y=2*x
y =
     20      4    NaN    Inf
>> z=y+2
z =
     22      6    NaN    Inf
>> w=(z==nan)
w =
      0      0      0      0
>> t=(z~=nan)
t =
      1      1      1      1
>>
```

上面的第 1 和第 2 计算式对 NaN 输入给出 NaN 结果。当 NaN 与 NaN 相比较时，(z==nan)产生全部为 0 或假的结果，同时(z~ =nan)产生全部 1 或真值。于是，单个 NaNs 相互不相等。由于 NaNs 的这种特性，如表 8-4 所示，MATLAB 7.0 有一个内置逻辑函数 isnan 寻找 NaNs。继续在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> g=isnan(z)
g =
      0      0      1      0
>> g1=isnan(y)
g1 =
      0      0      1      0
>>
```

这个函数用 find 函数能找出 NaNs 的下标值。继续在命令窗口中输入如下命令，并按 Enter 键确认。

```
>>   i=find(isnan(x))
```

```
i =
       3
>>
```

8.4.2　空矩阵的处理

当 NaNs 数学上由 IEEE 标准充分定义时，空矩阵由 MATLAB 7.0 的生成器确定，并有它自己的特性。空矩阵是简单的，它们是 MATLAB 7.0 大小为零的变量。

例 8-5　MATLAB 7.0 对空矩阵的操作。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> a=size([])
a =
       0       0
>> b=ones(4,0)
b =
    Empty matrix: 4-by-0
>> size(b)
ans =
       4       0
>> length(b)
ans =
       0
>>
```

上边的程序产生了两个空矩阵。但是在科学计算中，让一个空数组在任何维数上的长度为 0 有时具有特别的意义。当没有其他合适的结果时，在 MATLAB 7.0 中的许多函数返回空矩阵。最普通的例子是函数 find。继续在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> A=magic(4)
A =
    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
>> x=find(A==20)
x =
    Empty matrix: 0-by-1
>>
```

在上边的程序段中，由于魔术矩阵 x 中没有包含等于 20 的值，所以没有返回下标。返回一个空矩阵。为了测试空结果，MATLAB 7.0 提供了逻辑函数 isempty。继续在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> t=isempty(x)
t =
        1
>>
```

返回值为 1，可见 x 确实是一个空矩阵。

在 MATLAB 7.0 里，空矩阵不等于任何非零矩阵(或标量)，用户可以由下边的程序段得出结论。继续在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> y=[]
y =
        []
>> a=(y==0)
a =
        []
>>
```

返回的也是一个空矩阵，说明矩阵 y 没有元素等于 0。

8.5　各种运算符的优先级

同其他高级语言一样，MATLAB 7.0 语言对各种运算的优先级别做了规定。在进行各种运算时，遵守的一般规则是较高优先级的运算符的计算高于较低优先级的计算，相同优先级的运算符的计算遵从由左到右的原则。表 8-5 列出了 MATLAB 7.0 语言中各种运算级别的优先级。

表 8-5　运算符及其优先级

优　先　级	运　算　符
最高	()(小括号)
↓	.'(转置)　'(共轭转置)　.^(数组和数值乘方)　^(矩阵乘方)
↓	+(一元加法)　-(一元减法) ~(取反)
↓	.*(乘法)　*(矩阵乘法)　./(右除)　/(矩阵右除)　.\(左除)　\(矩阵左除)
↓	+(加法) -(减法)
↓	:(冒号)
↓	< (小于)　<=(小于或等于)　>(大于)　>=()大于或等于　==(等于)　~=()不等于
↓	& (逻辑与)
最低	|　(逻辑或)

下面举例予以说明。

例 8-6　比较各种运算符号的优先级别。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> x=5;y=3;z=2;
>> X=ones(3);
>> Y =magic(3);
>> Z=zeros(3),
>> a=x^2*(X'+Y)+z
a =
    227     52    177
    102    152    202
    127    252     77
>>
```

上边的程序段中，先进行小括号中间的运算，再计算 x^2，然后将两部分的结果相乘，最后再与 z 相加。继续在命令窗口中输入如下命令，并按 Enter 键确认。

```
>>b=Y&Z+x
b =
       1       1       1
       1       1       1
       1       1       1
>>
```

上边的程序段中，首先计算 Z 和 x 的加法运算，再与 Y 进行逻辑与运算。继续在命令窗口中输入如下命令，并按 Enter 键确认。

```
>>b=Y~=Z+X>=Y*x
b =
       0       0       0
       0       0       0
       0       0       0
>>
```

上边的程序段中，先计算 Y 和 x 的乘法，然后计算 Z 和 X 的加法，再进行逻辑运算。

8.6　习　　题

1. 讨论"=="和"="的区别，它们分别有什么用途？

2. 判断下面 MATLAB 7.0 语句的结果。

(1) 8<9

(2) 8<=9

(3) 8==9

(4) 8>0

(5) 8<=8

(6) 'A'<'b'

3. 设$a=20$，$b=-5$，$c=10$，$d=0$，试判断下列表达式的值。

(1) a>b

(2) a>c

(3) a>b&c>d

(4) a==c

(5) a&b>c

(6) ~~d

4. 设矩阵a、b、c和d的定义如下。

$a=[2]$； $b=\begin{bmatrix}2-1\\1-4\end{bmatrix}$； $c=\begin{bmatrix}2-11\\2-0\end{bmatrix}$； $d=\begin{bmatrix}1-1\\0-9\end{bmatrix}$。试对它们进行如下操作。

(1) ~(a>b)

(2) a>c&b>c

(3) c<=d

5. 设a、b、c和d的定义如下。

$a=-2$，$b=-5$，$c=150$，$d=0$，试分析下面算式的运算步骤并得出最终结果。

(1) a*b^2>a*c

(2) d|b>a

(3) (d|b)>a

6. 设a、b、c和d的定义如下。

$a=-2$，$b=-5$，$c=0$，$d=$'*Test*'，试分析如下算式的结果。

(1) isinf(a/b)

(2) isinf(a/c)

(3) a>b&ischar(d)

(4) isempty©

第9章　符号运算

在以前的 MATLAB 版本中，符号运算的功能比较薄弱，用户在解决比较复杂的符号运算时，单纯使用 MATLAB 往往无法求解，给广大的用户带来了很大的不便。从 MATLAB 5.3 开始，MATLAB 在符号运算方面的功能有了很大的提高，它采用全新的数据结构、面向对象编程和重载技术，使得符号计算和数值计算在形式和风格上浑然统一。而 MATLAB 7.0 提供了更为强大的符号运算功能，它专门提供了符号运算工具箱 Symbolic Math Toolbox，完全可以替代其他的符号运算专用计算语言，如 Maple 和 Mathematic 等。这样，用户只要掌握了 MATLAB，就可以解决数值运算、图形处理和符号运算 3 大基本运算。因此，MATLAB 语言成为各种数学语言中最受欢迎的语言。

在 MATLAB 7.0 语言中，既可以使用它本身开发的函数进行常用的符号运算，还可以通过 maple.m 和 map.m 两个接口和 Maple 相连。下面分别对这两种方法予以介绍。

(1) MATLAB 7.0 本身开发了许多专用于符号运算的函数，用户可以很方便地使用它们进行很多符号运算。MATLAB 7.0 根据不同的模块，将这些函数分成如下几类：

- 符号表达式和符号矩阵的操作
- 符号微积分
- 符号线性方程
- 符号微分方程

(2) 尽管 MATLAB 7.0 的符号运算功能很强大，但是一门优秀的语言往往能够博采众家之长，MATLAB 7.0 语言提供了和 MAPLE 语言的良好接口，通过 maple.m 和 map.m 两个专用的 M 文件来实现。这样，MATLAB 7.0 关于符号运算的能力就更强大了。

9.1　符号变量的生成和使用

上面简单回顾了一下字符型数据变量的生成，本节将介绍如何生成符号型数据变量。生成符号型数据变量要使用专门的函数 sym 和 syms。下面将分别予以介绍。

9.1.1　符号变量、符号表达式和符号方程的生成

1. 使用 sym 函数定义符号变量和符号表达式

sym 函数可以生成单个的符号变量，下面通过一个例子来介绍它的使用方法，并与其他的数据类型做比较。

例 9-1　sym 函数的使用方法。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> sqrt(2)
ans =
     1.4142
>> a=sqrt(sym(2))
 a =
 2^(1/2)
 >> double(a)
ans =
     1.4142
>> sym(2)/sym(5)
ans =
 2/5
>> 2/5+1/3
ans =
     0.7333
>> sym(2)/sym(5)+sym(1)/sym(3)
 ans =
 11/15
  >>
```

同时，使用 sym 函数也可以定义符号表达式，此时，有两种定义方法，一是使用 sym 函数将式中的每一个变量定义为符号变量；二是使用 sym 函数将整个表达式集体定义。但是，在使用第二种方法时，虽然也生成了与第一种方法相同的表达式，但是并没有将里边的变量也定义为符号变量。

例 9-2　使用 sym 函数定义符号表达式 ax^2+bx+c。

解：首先采用单个变量定义法，在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> a = sym('a');
>> b = sym('b') ;
>> c = sym('c');
>> x = sym('x');

>> f=a*x^2 + b*x + c
f =
a*x^2+b*x+c
>>
```

也可以采用整体定义法，此时，将整个表达式用单引号括起来，再用 sym 函数加以定义，继续在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> f = sym('a*x^2 + b*x + c')
f
a*x^2 + b*x + c
```

```
>> g=f^2+4*f-2
g =
 (a*x^2+b*x+c)^2+4*a*x^2+4*b*x+4*c-2
>>
```

2. 使用 syms 函数定义符号变量和符号表达式

syms 函数的功能比 sym 函数要更为强大，它可以一次创建任意多个符号变量。而且，syms 函数的使用格式也很简单，使用格式如下。

```
syms   var1 var2 vra3……
```

下面通过一个例子来介绍 syms 函数的使用方法。

例 9-3 使用 syms 函数定义字符变量和数组。

解：仍然定义例 6-2 中的标准表达式，在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> syms a b c x
>> f = sym('a*x^2 + b*x + c')
f =
a*x^2 + b*x + c
>> g=f^2+4*f-2
g =
 (a*x^2+b*x+c)^2+4*a*x^2+4*b*x+4*c-2
>>
```

3. 符号方程的生成

方程与函数的区别在于函数是一个由数字和变量组成的代数式，而方程则是由函数和等号组成的等式。在 MATLAB 7.0 语言中，生成符号方程的方法与使用 sym 函数生成符号函数类似，但是不能采用直接生成法生成符号函数。

例 9-4 使用 sym 函数生成符号方程。

```
>> %符号方程的生成
>> %使用 sym 函数生成符号方程
>> equation1=sym('sin(x)+cos(x)=1')
equation1 =
sin(x)+cos(x)=1
>>
```

9.1.2 符号变量的基本操作

1. findsym 函数用于寻找符号变量

该函数用于找出一个表达式中存在那些符号变量，例如，给定由符号变量定义的符号表达式 f 和 g，其中 $f = x^n$，$g = \sin(a \times t + b)$。那么，使用 $findsym(f)$ 和 $findsym(g)$

可以分别找出两个表达式中的符号变量，此外，对于任意表达式 s，使用 *findsym*(s,n) 可以找出表达式 s 中 n 个与 x 接近的变量。

例 9-5　使用 findsym 函数寻找符号表达式中的符号变量。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> syms a alpha   b x1 y
>> findsym(alpha+a+b)
ans =
a, alpha, b
>> findsym(cos(alpha)*b*x1 + 14*y,2)
ans =
x1,y
>> findsym(y*(4+3*i) + 6*j)
ans =
y
>>
```

2. 任意精确度的符号表达式

MATLAB 7.0 提供了 digits 和 vpa 这两个函数来实现任意精度的符号运算。

(1) digits 函数设定所用数值的精度。

- 单独使用 digits 命令将在命令窗口中显示当前设定的数值精度。
- digits(D) 命令将设置数值的精度为 D 位。其中 D 为一个整数，或者是一个表示数的字符型变量或符号变量。
- D = digits 命令也是在命令窗口中返回当前设定数值精度，其中 D 是一个整数。

例 9-6　使用 digits 函数设置数值精度。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> digits
Digits = 32
>>
```

此时，由输出结果可以知道当前的数值精度为 32 位。

继续在命令窗口中输入如下命令，并按 Enter 键确认。

```
>>digits(100)
>>
```

此时，命令窗口没有任何反应，但是，系统内部已经将数值精度设定位 100 位。

继续在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> d=digits
d =
   100
```

```
>>
```

可见，此时，数值精度已经设定为100位。

(2) vpa函数进行可控精度运算

- R = vpa(S) 命令将显示符号表达式 S 在当前精度 D 下的值。其中 D 是使用 digits 函数设置的数值精度。
- vpa (S,D)命令显示符号表达式 S 在精度 D 下的值，这里的 D 不是当前的精度值，而是临时使用 digits 函数设置为 D 位精度。

例 9-7 使用 vpa 函数进行可控精度运算。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> r=vpa(pi)
r =
3.1415926535897932384626433832795
>>
>> q=vpa(hilb(2))
q =
[                                  1., .50000000000000000000000000000000]
[.50000000000000000000000000000000, .33333333333333333333333333333333]
>>
```

可见，此时系统的默认精度值为32位，所显示的值都有32位小数。

继续在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> p=vpa(pi,1000)
p =
3.1415926535897932384626433832795028841971693993751058209749445923078164062862089986
2803482534211706798214808651328230664709384460955058223172535940812848111745028410270
1938521105559644622948954930381964428810975665933446128475648233786783165271201909145 6
4856692346034861045432664821339360726024914127372458700660631558817488152092096282925 4
0917153643678925903600113305305488204665213841469519415116094330572703657595919530921 8
6117381932611793105118548074462379962749567351885752724891227938183011949129833673362 4
4065664308602139494639522473719070217986094370277053921717629317675238467481846766940 5
1320005681271452635608277857713427577896091736371787214684409012249534301465495853710 5
0792279689258923542019956112129021960864034418159813629774771309960518707211349999998 3
7297804995105973173281609631859502445945534690830264252230825334468503526193118817101 0
0031378387528865875332083814206171776691473035982534904287554687311595628638823537875 9
375195778185778053217122680661300192787661119590921642019 9
>>
 >> q=vpa(hilb(2),6)
q =
[        1., .500000]
[.500000, .333333]
>>
```

此时，用户可以看出第一条命令将 pi 的精度定为 1000 位，而第二条命令将矩阵的精度定为 6 位。

3. 数值型变量与符号型变量的转换形式

对于任意数值型变量 t，使用 sym 函数可以将其转换为 4 种形式的符号变量，分别为有理数形式：*sym(t)* 或 *sym(t,'r')*、浮点数形式 *sym(t,'f')*、指数形式 *sym(t,'e')* 和数值精度形式 *sym(t,'d')*。下面举例予以说明。

例 9-8 使用 sym 函数进行数值型变量与符号型变量的转换。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> t=0.1
t =
     0.1000
>> sym(t)             %有理数形式
 ans =
 1/10
>> sym(t,'r')          %有理数形式
 ans =
 1/10
>> sym(t,'f')          %浮点数形式
ans =
 '1.999999999999a'*2^(-4)
>> sym(t,'e')          %指数形式
ans =
1/10+eps/40
>> sym(t,'d')          %数值精度形式
ans =
.10000000000000000555111512312578
>>
```

在 MATLAB 7.0 中默认的精度是 32 位，因此，上边的显示值也是具有 32 位精度。如果用户想改变显示的精度，可以使用 digits 函数进行设置。继续在命令窗口中输入如下命令，并按 Enter 键确认。

```
>>digits(7)
>> sym(t,'d')
ans =
 .1000000
>>
```

此外，也可以使用上述方法将数值型矩阵转换为符号型矩阵。注意此时只能将其转换为有理数形式，如果用户想转换为其他 3 种类型，MATLAB 7.0 将给出错误的提示。继续在命令窗口中输入如下命令，并按 Enter 键确认。

```
>>A=hilb(4)
A =
     1.0000    0.5000    0.3333    0.2500
     0.5000    0.3333    0.2500    0.2000
     0.3333    0.2500    0.2000    0.1667
     0.2500    0.2000    0.1667    0.1429
>> A=sym(A)
A =
 [   1, 1/2, 1/3, 1/4]
[ 1/2, 1/3, 1/4, 1/5]
[ 1/3, 1/4, 1/5, 1/6]
[ 1/4, 1/5, 1/6, 1/7]
>> A=sym(A,'d')                                %错误的提示
??? Error using ==> sym.sym
Second argument d not recognized.
>> A=sym(A,'e')
??? Error using ==> sym.sym
Second argument e not recognized.
>> A=sym(A,'f')
??? Error using ==> sym.sym
Second argument f not recognized.
>>
```

9.1.3 符号表达式(符号函数)的操作

用户可以对符号表达式进行各种操作，包括四则运算、合并同类型、多项式分解和简化等。下面简单予以说明。

1. 符号表达式的四则运算

符号表达式也与通常的算术式一样，可以进行四则运算。

例 9-9 符号表达式的四则运算。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> syms x y a b
>> fun1=sin(x)+cos(y)
fun1 =
sin(x)+cos(y)
>> fun2=a+b
fun2 =
a+b
>> fun1+fun2
ans =
sin(x)+cos(y)+a+b
```

```
>>fun1*fun2
ans =
(sin(x)+cos(y))*(a+b)
>>
```

2. 合并符号表达式的同类项

在 MATLAB 7.0 语言中，使用 collect 函数来合并符号表达式的同类项，其使用格式如下：

- *collect*(*S*,*v*) 命令将符号矩阵 *S* 中所有同类项合并，并以 *v* 为符号变量输出。
- *collect*(*S*) 命令使用 findsym 函数规定的默认变量代替上式中的 *v*。

例 9-10　符号多项式同类项的合并。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> syms x y
>> collect(x^2*y + y*x - x^2 - 2*x)
 ans =
 (y-1)*x^2+(y-2)*x
>> f = -1/4*x*exp(-2*x)+3/16*exp(-2*x);
>> collect(f)
 ans =
 -1/4*x*exp(-2*x)+3/16*exp(-2*x)
 >>
```

3. 符号多项式的因式分解

在 MATLAB 7.0 语言中，使用 horner 函数进行符号多项式的合并，其使用格式为，*horner*(*P*) 命令将符号表达式 *P* 进行因式分解。

例 9-11　符号多项式的分解。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
% 使用 homor 函数进行多项式分解
>> syms x
>> fun1=2*x^3+2*x^2-32*x+40
fun1 =
2*x^3+2*x^2-32*x+40
>> horner(fun1)
ans =
40+(-32+(2+2*x)*x)*x
>>   fun2=x^3-6*x^2+11*x-6
 fun2 =
 x^3-6*x^2+11*x-6
>> horner(fun2)
ans =
 -6+(11+(-6+x)*x)*x
>>
```

4. 符号表达式的简化

在 MATLAB 7.0 语言中，使用 simplify 函数和 simple 函数进行符号表达式的简化。下面对它们的使用方法进行介绍。

(1) simplify 函数的使用。

simplify(*S*) 命令将符号表达式 *S* 中的每一个元素都进行简化，该函数的缺点是即使多次运用 simplify 也不一定不能得到最简形式。

例 9-12　使用 simplify 函数进行符号函数的简化。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> syms x
>> fun1=(1/x+7/x^2+12/x+8)^(1/3)
fun1 =
(13/x+7/x^2+8)^(1/3)
>> sfy1=simplify(fun1)
sfy1 =
((13*x+7+8*x^2)/x^2)^(1/3)
>> sfy2=simplify(sfy1)
sfy2 =
((13*x+7+8*x^2)/x^2)^(1/3)
>> simplify(sin(x)^2 + cos(x)^2)
 ans =
1
>>
```

(2) simple 函数的使用。

用 simple 函数对符号表达式进行简化，该方法比使用 simplify 函数要简单，所得的结果也比较合理。其使用格式如下。

- *simple*(*S*) 命令使用多种代数简化方法对符号表达式 *S* 进行简化，并显示其中最简单的结果。
- [*R*, *how*] = *simple*(*S*) 命令在返回最简单的结果的同时，返回一个描述简化方法的字符串 *how*。

例 9-13　使用 simple 函数进行符号函数的简化。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> s=2*cos(x)^2-sin(x)^2 ;
```

定义表达式 *S* 后，先使用 *simple*(*S*) 命令对其进行简化。

```
>> simple(s)
 simplify:
 3*cos(x)^2-1
```

```
radsimp:
  2*cos(x)^2-sin(x)^2
combine(trig):
3/2*cos(2*x)+1/2
factor:
2*cos(x)^2-sin(x)^2
expand:
  2*cos(x)^2-sin(x)^2
combine:
3/2*cos(2*x)+1/2
convert(exp):

2*(1/2*exp(i*x)+1/2/exp(i*x))^2+1/4*(exp(i*x)-1/exp(i*x))^2
convert(sincos):
  2*cos(x)^2-sin(x)^2
convert(tan):
  2*(1-tan(1/2*x)^2)^2/(1+tan(1/2*x)^2)^2-4*tan(1/2*x)^2/(1+tan(1/2*x)^2)^2
collect(x):
2*cos(x)^2-sin(x)^2
mwcos2sin:
2-3*sin(x)^2
ans =
3*cos(x)^2-1
>>
```

下面再应用$[R,how]=simple(S)$命令对相同的表达式进行简化，用户可以从中对比两个命令的区别，继续在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> [R,how] = simple(s)
R =
3*cos(x)^2-1
how =
simplify
>>
```

5. subs 函数用于替换求值

使用 subs 函数可以将符号表达式中的字符型变量用数值型变量替换，其使用方法如下。

- $subs(S)$命令将符号表达式S中的所有符号变量用调用函数中的值或是 MATLAB 7.0 工作区间的值代替。
- $subs(S,new)$命令将符号表达式S中的自由符号变量用数值型变量或表达式new替换。例如用户想求表达式$f=2x^2-3x+1$当$x=2$时的值，可以使用$subs(f,2)$。
- $subs(S,old,new)$命令将符号表达式S中的符号变量old用数值型变量或表达式new替换。

例 9-14 使用 subs 函数进行替换求值操作。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> syms x y
f = x^2*y + 5*x*sqrt(y)
f =
x^2*y+5*x*y^(1/2)
>> subs(f, x, 3)
ans =
 9*y+15*y^(1/2)
>> subs(f, y, 3)
ans =
3*x^2+5*x*3^(1/2)
>>
```

如果用户没有指定被替换的符号变量，那么 MATLAB 7.0 将按如下规则选择默认的替换变量，对于单个字母的变量， MATLAB 7.0 选择在字母表中与 x 最接近的字母，如果有两个变量离 x 一样近， MATLAB 7.0 将选择字母表中靠后的那个。因此，在上边的程序段中，$subs(f,x,3)$ 与 $subs(f,3)$ 的返回值是相同的，用户可以使用 findsym 函数寻找默认的替换变量。如下边的程序段所示，继续在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> syms s t
>> g = s + t;
>> findsym(g,1)
ans =
t
>>
```

以上部分的程序段进行了单个变量的替换，使用 subs 函数也可以进行多个变量的替换，如下边的程序段所示，继续在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> subs(cos(a)+sin(b),{a,b},{sym('alpha'),2})
ans =
 cos(alpha)+sin(2)
 >>
```

同时，也可以使用矩阵作为替换变量，用来替换符号表达式中的符号变量，继续在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> syms t
>> subs(exp(a*t),'a',-magic(2))
 ans =
[     exp(-t), exp(-3*t)]
[ exp(-4*t), exp(-2*t)]
>>subs(x*y,{x,y},{[0 1;-1 0],[1 -1,-2 1]})
```

```
ans =
      0     -1
      2      0
>>
```

6. 反函数的运算

反函数运算是符号运算的重要组成部分，在 MATLAB 7.0 语言中，使用 finverse 函数来实现对符号函数的反函数运算。其使用格式如下。

- $g = finverse(f)$ 命令用于求函数 f 的反函数，其中 f 为一符号表达式，x 为单变量，函数 g 也是一个符号函数，且满足 $g(f(x)) = x$。
- $g = finverse(f, v)$ 命令所返回的符号函数表达式的自变量是 v，这里 v 是一个符号变量，且是表达式的向量变量。而 g 的表达式要求满足 $g(f(x)) = v$。当 f 包括不止一个变量时最好使用该命令。

例 9-15　使用 finverse 函数进行反函数运算。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> syms x y
>> f = x^2+y]
>> f = x^2+y
f =
x^2+y
>> finverse(f,y)
ans =
-x^2+y
>> finverse(f)
Warning: finverse(x^2+y) is not unique.
> In C:\MATLAB 7 06p5\toolbox\symbolic\@sym\finverse.m at line 43
ans =
(-y+x)^(1/2)
>>
```

此时，由于用户没有指明自变量，MATLAB 7.0 语言将给出警告信息，并且以 x 为默认变量给出结果。

再如，求函数 $f = x^2$ 的反函数的程序段如下。

```
>> syms x
>> f=x^2
f =
x^2
>> g=finverse(f)
Warning: finverse(x^2) is not unique.
> In C:\MATLAB 7.06p5\toolbox\symbolic\@sym\finverse.m at line 43
g
```

```
x^(1/2)
>>
```

可见，由于函数 $f=x^2$ 的反函数不惟一，MATLAB 7.0 语言将给出警告信息，并且以 x 默认为正值给出反函数。

我们可以验证 finverse 函数的正确性，及验算 $g(f(x))$ 是否等于 x。程序段如下：

```
>> fg=simple(compose(g,f))
fg =
x
>>
```

以上的程序使用了 compose 函数和 simple 函数。下面将对 compose 函数的应用予以详细介绍。

7. 复合函数的运算

在科学计算中，经常要遇到求解复合函数的情况，比如函数 $z=f(y)$，而该函数的自变量 y 又是另外一个函数，$y=g(x)$，也就是 $z=f(g(x))$，此时，求 z 对 x 的函数的过程就是求解复合函数的过程。

在 MATLAB 7.0 语言中，提供了专门用于进行复合函数运算的函数 compose。它的使用方法如下：

- $compose(f,g)$ 命令返回当 $f=f(x)$ 和 $g=g(y)$ 时的复合函数 $f(g(y))$。这里 x 是为 findsym 定义的 f 的符号变量，y 是为 findsym 定义的 g 的符号变量。
- $compose(f,g,z)$ 命令返回 $f=f(x)$ 和 $g=g(y)$ 时的复合函数 $f(g(z))$，返回的函数以 z 为自变量。这里 x 是为 findsym 定义的 f 的符号变量，y 是为 findsym 定义的 g 的符号变量。
- $compose(f,g,x,z)$ 命令返回复合函数 $f(g(z))$，这里 x 是函数 f 的独立的变量。也就是说，例如若 $f=\cos(x/t)$，那么 $compose(f,g,x,z)$ 命令将返回 $\cos(g(z)/t)$，而 $compose(f,g,t,z)$ 命令将返回 $\cos(x/g(z))$。
- $compose(f,g,x,y,z)$ 命令返回 $f(g(z))$ 并使得 x 为函数 f 的独立变量，y 是函数 g 的独立变量。例如若 $f=\cos(x/t)$ 并且 $g=\sin(y/u)$，那么 $compose(f,g,x,y,z)$ 命令将返回 $\cos(\sin(z/u)/t)$ 而 $compose(f,g,x,u,z)$ 命令将返回 $\cos(\sin(y/z)/t)$。

例 9-16 复合函数的运算。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> syms x y z t u
>> f = 1/(1 + x^2)
f =
1/(1+x^2)
>> g = sin(y)
```

```
g =
sin(y)
>>   h = x^t
h =
x^t
>>   p = exp(-y/u)
p =
exp(-y/u)
>> compose(f,g)
ans =
1/(1+sin(y)^2)
  >> compose(f,g,t)
ans =
1/(1+sin(t)^2)
>> compose(h,g,x,z)
ans =
sin(z)^t
>> compose(h,g,t,z)
ans =
x^sin(z)
>>compose(h,p,x,y,z)
ans =
exp(-z/u)^t
>>compose(h,p,t,u,z)
ans =
x^exp(-y/z)
>>
```

9.2　符号矩阵的生成和运算

本节介绍在 MATLAB 7.0 语言中如何生成符号矩阵及其运算。

9.2.1　符号矩阵的生成

在 MATLAB 7.0 语言中，符号矩阵的生成与数值矩阵的相关操作很相似，但是要用到符号定义函数 sym，本小节将介绍怎样使用该函数生成符号矩阵。

1. 使用 sym 函数直接生成符号矩阵

该方法简单使用，用户在学习了前边的章节之后，就可以用与生成数值矩阵相同的方法直接生成符号矩阵。矩阵元素之间可以使用空格或逗号分隔；各符号表达式的长度可以不同；矩阵元素可以是任意的符号函数。

例 9-17 使用 sym 函数直接生成符号矩阵。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> a1=sym('[1/3 2/3 5/7;9/11 11/13 13/17;17/19 19/23 23/29]')
a1 =
[   1/3,     2/3,     5/7]
[   9/11, 11/13, 13/17]
[ 17/19, 19/23, 23/29]
>>
```

在上面的程序中，使用了空格作为矩阵元素之间的分隔，且各符号表达式的长度相同。继续在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> a1=sym('[1/3,0.2+sqrt(2),pi;2/7,sin(x),cos(x),log(x),sin(x)^2,sin(22*x),exp(x)]')
a1 =
[1/3,0.2+sqrt(2),pi]
[2/7,sin(x),cos(x),log(x)]
[sin(x)^2,sin(22*x),exp(x)]
>>
```

在上面的程序中，使用了逗号作为矩阵元素之间的分隔，且各符号表达式的长度不相同。

2. 用生成子矩阵的方法生成符号矩阵

与字符串矩阵的直接输入法相似，可以采用直接输入字符串的办法生成符号矩阵，该方法不需要调用 sym 函数，但要保证同一列的元素具有相同的长度。因此，用户在使用中要注意。当要输入的字符串长度不一致时，用户可以在较短字符串的前边或后边加上空格符补充。

例 9-18 用生成子矩阵的方法生成符号矩阵。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> a=['[100,cos(x)]';'[1/s,x        ]']
a =
[100,cos(x)]
[1/s,x        ]
>>
```

3. 由数值矩阵转换为符号矩阵

在 MATLAB 7.0 语言中，数值型变量和符号型变量是两种不同的数据类型，因此，在 MATLAB 7.0 中，分属于这两个数据类型的变量不能直接进行运算，系统首先将自动在 MATLAB 7.0 的工作区间将数值型变量转换为符号型变量，然后再进行计算。用户也可以通过使用 sym 函数先将数值型变量转换为符号型变量，然后再进行计算。

例 9-19　使用 sym 函数将数值矩阵转换为符号矩阵。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> M=[30 1 1 1;6 1 5 9;9 8 25 4;32 45 62 0]
M =
    30     1     1     1
     6     1     5     9
     9     8    25     4
    32    45    62     0
>> S=sym(M)
S =
[ 30,  1,  1,  1]
[  6,  1,  5,  9]
[  9,  8, 25,  4]
[ 32, 45, 62,  0]
>>
```

可见，对于矩阵元素为整数的情况，使用 sym 函数之后，矩阵形式上并没有太大的变化。但是，在 MATLAB 7.0 的工作区间内，系统已经生成了一个新的矩阵，其数据类型为符号型，如图 9-1 所示。

下面的程序演示当矩阵元素为分数或无理数时，使用 sym 函数之后的效果。继续在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> M1=[0.3 0.33 0.333 1/3;3.14 3.142 3.1416 pi;log(2) log(3) log(5) log(7);sin(1) cos(1) tan(1)
atan(1)]
M1 =
    0.3000    0.3300    0.3330    0.3333
    3.1400    3.1420    3.1416    3.1416
    0.6931    1.0986    1.6094    1.9459
    0.8415    0.5403    1.5574    0.7854
>> S1=sym(M1)
S1 =
[                  3/10,                  33/100,                  333/1000,                  1/3]
[                157/50,                1571/500,                 3927/1250,                   pi]
[ 6243314768165359*2^(-53), 4947709893870346*2^(-52), 7248263982714163*2^(-52),
8763600222181975*2^(-52)]
[ 7579296827247854*2^(-53), 4866610526750348*2^(-53), 7013940848419750*2^(-52),
pi/4]
>>
```

此时系统又在工作区间生成了新的矩阵 M1 和 S1，如图 9-2 所示。

由程序结果可见，当矩阵元素以分数或是浮点数形式存在时，当其转换为符号矩阵后，系统将以最接近原始数值的精确有理数形式给出结果。

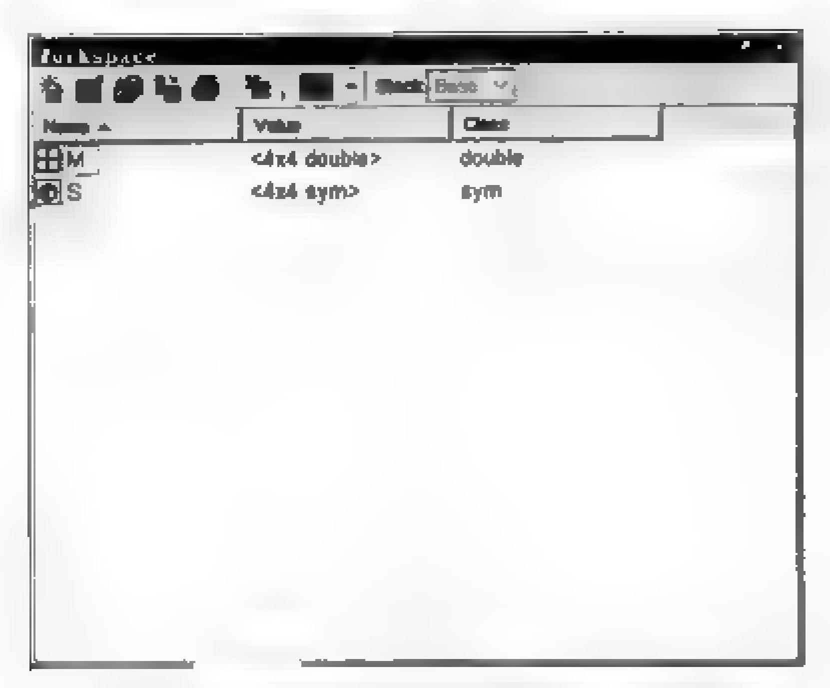


图9-1 使用sym函数将数值矩阵转换为符号矩阵1

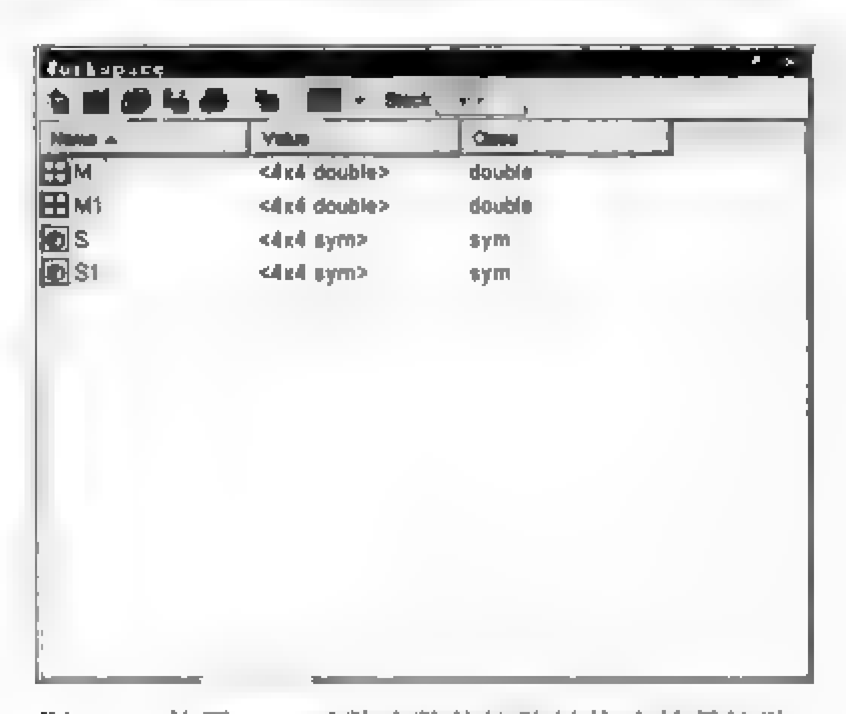


图9-2 使用sym函数将数值矩阵转换为符号矩阵2

9.2.2 符号矩阵及符号数组的运算

前面的章节已经介绍了一些关于符号表达式的操作。但是，由于MATLAB 7.0语言是基于矩阵(或数组)操作的一门语言，因此，具体到符号矩阵和符号数组，MATLAB 7.0提供了更为详细的关于进行符号矩阵运算的函数。

1. 符号矩阵的四则运算

- $A+B$ 和 $A-B$ 命令可以实现符号阵列的加法与减法。若 A 与 B 为同型阵列时，$A+B$、$A-B$ 分别对对应分量进行加减；若 A 与 B 中至少有一个为标量，则把标量扩大为与另外一个同型的阵列，再按对应的分量进行加减。
- $A*B$ 命令可以实现符号矩阵的乘法。$A*B$ 为线性代数中定义的矩阵乘法。按乘法定义要求必须有矩阵 A 的列数等于矩阵 B 的行数（若 $A_{n*k} * B_{k*m} = (a_{ij})_{n*k} .* (b_{ij})_{n*m} = C_{n*m} = (c_{ij})_{n*m}$ ，则 $c_{ij} = \sum_{s=1}^{k} a_{is} * b_{sj}$ ，

$i=1,2,\ldots,n; j=1,2,\ldots,m$)或者至少有一个为标量时，方可进行乘法操作，否则系统将返回一出错信息。

- $A\backslash B$ 命令可以实现矩阵的左除法。$X=A\backslash B$ 为符号线性方程组 $A*X=B$ 的解。需要指出的是，$A\backslash B$ 近似地等于 $inv(A)*B$。若 X 不存在或者不惟一，则产生一警告信息。矩阵 A 可以是矩形矩阵(即非正方形矩阵)，但此时要求方程组必须是相容的。
- A/B 命令可以实现矩阵的右除法。$X=A/B$ 为符号线性方程组 $X*A=B$ 的解。需要指出的是，B/A 粗略地等于 $B*inv(A)$。若 X 不存在或者不惟一，则提示警告信息。矩阵 A 可以是矩形矩阵(即非正方形矩阵)，但此时要求方程组必须是相容的。

例 9-20　矩阵的四则运算。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> m=sym('[x,x^2,x*2,1/x]')
m =
[     x, x^2, x*2, 1/x]
>> n=sym('[2*x,y,x,x^2]')
n =
[ 2*x,     y,     x, x^2]
>> m+n
ans =
[       3*x,     x^2+y,         3*x, 1/x+x^2]
>> m-n
ans =
[         -x,     x^2-y,            x, 1/x-x^2]
>>
```

以上程序段演示了符号矩阵的加法运算和减法运算，下面的程序段将演示符号矩阵的乘法和除法运算。

继续在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> a=m'*n
a =
[     2*x*conj(x),        conj(x)*y,         x*conj(x),     conj(x)*x^2]
[ 2*conj(x)^2*x,     conj(x)^2*y,     conj(x)^2*x, conj(x)^2*x^2]
[     4*x*conj(x),     2*conj(x)*y,     2*x*conj(x), 2*conj(x)*x^2]
[     2/conj(x)*x,     1/conj(x)*y,     1/conj(x)*x, 1/conj(x)*x^2]
>> b=m\n
b =
[         0,         0,         0,         0]
[         0,         0,         0,         0]
[         0,         0,         0,         0]
[ 2*x^2,     y*x,     x^2,     x^3]
```

```
>> c=m/n
Warning. System is inconsistent. Solution does not exist.
> In C:\MATLAB 7 06p5\toolbox\symbolic\@sym\mldivide.m at line 29
  In C:\MATLAB 7 06p5\toolbox\symbolic\@sym\mrdivide.m at line 24

c =
Inf
>>
```

由程序结果可见，由于m/n的结果不存在或者不惟一，所以系统提示错误信息，并将其值定为Inf。

2. 符号数组的四则运算

- $A.*B$ 命令用于符号数组的乘法运算。$A.*B$ 为按参量 A 与 B 对应的分量进行相乘。A 与 B 必须为同型阵列，或至少有一个为标量。即：$A_{n*m}.*B_{n*m}=(a_{ij})_{n*m}.*(b_{ij})_{n*m}=C_{n*m}=(c_{ij})_{n*m}$，则 $c_{ij}=\sum_{s=1}^{k}a_{ij}*b_{ij}$，$i=1,2,\ldots,n; j=1,2,\ldots,m$。
- $A./B$ 命令用于数组的右除法运算。$A./B$ 为按对应的分量进行相除。若 A 与 B 为同型阵列时，$A_{n*m}./B_{n*m}=(a_{ij})_{n*m}./(b_{ij})_{n*m}=C_{n*m}=(c_{ij})_{n*m}$，则 $c_{ij}=\sum_{s=1}^{k}a_{ij}*b_{ij}$，$i=1,2,\ldots,n; j=1,2,\ldots,m$。若A与B中至少有一个为标量，则把标量扩大为与另外一个同型的阵列，再按对应的分量进行操作。
- $A.\backslash B$ 命令用于数组的左除法运算。$A.\backslash B$ 为按对应的分量进行相除。若 A 与 B 为同型阵列时，$A_{n*m}.\backslash B_{n*m}=(a_{ij})_{n*m}.\backslash(b_{ij})_{n*m}=C_{n*m}=(c_{ij})_{n*m}$，则 $c_{ij}=\sum_{s=1}^{k}a_{ij}*b_{ij}$，$i=1,2,\ldots,n; j=1,2,\ldots,m$。若 A 与 B 中至少有一个为标量，则把标量扩大为与另外一个同型的阵列，再按对应的分量进行操作。

例9-21 符号数组的四则运算。

解：在命令窗口中输入如下命令，并按Enter键确认。

```
>> q=sym('[3,4,9,6;x,y,z,w;a,b,c,d]')
q =
[ 3, 4, 9, 6]
[ x, y, z, w]
[ a, b, c, d]
>> p=sym('[x,1/x,x^2,x^3;a,b,c,d;5,2,3,6]')
p =
[    x, 1/x, x^2, x^3]
[    a,   b,   c,   d]
```

```
[    5,    2,    3,    6]
>> r=q*p
??? Error using ==> sym/mtimes
Inner matrix dimensions must agree.
>>
```

由于对矩阵进行乘法操作时，第一个矩阵的列必须和第二个矩阵的行的元素个数相同，因此系统提示出错信息。

继续在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> q.*p
ans =
[    3*x,    4/x, 9*x^2, 6*x^3]
[    x*a,    y*b,    z*c,    w*d]
[    5*a,    2*b,    3*c,    6*d]
>> q./p
ans =
[    3/x,    4*x, 9/x^2, 6/x^3]
[    x/a,    y/b,    z/c,    w/d]
[ 1/5*a, 1/2*b, 1/3*c, 1/6*d]
>> q.\p
ans =
[    1/3*x,    1/4/x, 1/9*x^2, 1/6*x^3]
[        a/x,        b/y,        c/z,        d/w]
[        5/a,        2/b,        3/c,        6/d]
>>
```

3. 矩阵和数组的逆运算

- A' 命令可以实现矩阵的 Hermition 转置。若 A 为复数矩阵，则 A' 为复数矩阵的共轭转置。即，若 $A=(a_{ij})=(x_{ij}+i*y_{ij})$，则 $A'=(a_{ji})=(\overline{a_{ij}})=(x_{ij}-i*y_{ij})$。
- $A.'$ 数组转置。$A.'$ 为真正的矩阵转置，其没有进行共轭转置。

例 9-22　矩阵和数组的逆运算。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
q =
[ 3, 4, 9, 6]
[ x, y, z, w]
[ a, b, c, d]
>> q'
ans =
[            3, conj(x), conj(a)]
[            4, conj(y), conj(b)]
[            9, conj(z), conj(c)]
```

```
[          6, conj(w), conj(d)]
>>
```

以上所求为矩阵 q 的 Hermition 转置矩阵，由于 x、y、z、w、a、b、c 和 d 都是符号变量，系统无法给出具体值，只能用 conj(x)等值给出。

继续在命令窗口中输入如下命令，并按 Enter 键确认。

```
>>q.'
ans =
[ 3, x, a]
[ 4, y, b]
[ 9, z, c]
[ 6, w, d]
>>
```

该段程序所求值即为 q 矩阵的倒置。

4. 矩阵和数组的幂运算

- *A*^*B* 命令可以实现矩阵的幂运算。计算矩阵 *A* 的整数 *B* 次方幂。若 *A* 为标量而 *B* 为方阵，*A*^*B* 用方阵 *B* 的特征值与特征向量计算数值。若 *A* 与 *B* 同时为矩阵，则返回一错误信息。
- *A*.^*B* 命令可以实现数组的幂运算。*A*.^*B* 为按 *A* 与 *B* 对应的分量进行方幂计算。若 *A* 与 *B* 为同型阵列时，$A_{n*m}.\wedge B_{n*m}=(a_{ij})_{n*m}.\wedge(b_{ij})_{n*m}=C_{n*m}=(c_{ij})_{n*m}$，则 $(c_{ij})=a_{ij}\wedge b_{ij}$，$i=1,2,\ldots,n; j=1,2,\ldots m$。若 *A* 与 *B* 中至少有一个为标量，则把标量扩大为与另外一个同型的阵列，再按对应的分量进行操作。

例 9-23 矩阵和数组的幂运算。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> p=sym('[x,1/x,x^2,x^3;a,b,c,d;5,2,3,6]')
p =
[    x, 1/x, x^2, x^3]
[    a,    b,    c,    d]
[    5,    2,    3,    6]
>> q=sym('[3,4,9,6;x,y,z,w;a,b,c,d,1 3 5 7]')
q =
[ 3, 4, 9, 6]
[ x, y, z, w]
[ a, b, c, d]
[ 1, 3, 5, 7]
>> q^2
ans =
[        15+4*x+9*a,        30+4*y+9*b,        57+4*z+9*c,        60+4*w+9*d]
```

```
[     3*x+y*x+z*a+w, 4*x+y^2+z*b+3*w, 9*x+y*z+z*c+5*w, 6*x+y*w+z*d+7*w]
[     3*a+b*x+c*a+d, 4*a+y*b+c*b+3*d, 9*a+z*b+c^2+5*d, 6*a+b*w+c*d+7*d]
[          10+3*x+5*a,          25+3*y+5*b,          44+3*z+5*c,          55+3*w+5*d]
>> p^2
??? Error using ==> sym/mpower
Matrix must be square
>>
```

可见，由于p矩阵不是方阵，无法进行矩阵的幂运算，系统将提示出错警告。继续在命令窗口中输入如下命令，并按Enter键确认。

```
>>q ^2
ans =
[     9,   16,   81,   36]
[ x^2, y^2, z^2, w^2]
[ a^2, b^2, c^2, d^2]
[     1,     9,   25,   49]
>> p.^2
ans =
[     x^2, 1/x^2,     x^4,     x^6]
[     a^2,     b^2,     c^2,     d^2]
[       25,         4,         9,       36]
>>
```

5. 符号矩阵的秩

在MATLAB 7.0语言中，使用rank函数来求符号矩阵的秩，其使用格式如下。

$rank(A)$命令求出方阵A的线性不相关的独立行和列的个数。而$rank(A,tol)$命令则求出A中比tol值大的值的个数，在$rank(A)$命令中，默认$tol = \max(size(A)) * norm(A) * eps$。

例9-24　符号矩阵的秩。

解：在命令窗口中输入如下命令，并按Enter键确认。

```
>> a=sym('[1,1/x,x^2,xin(x),cos(x),tan(x);log(x),2,9]')
a =
[          1,        1/x,        x^2]
[ xin(x), cos(x), tan(x)]
[ log(x),           2,            9]
>> rank(a)
ans =
3
>>
```

6. 符号矩阵的逆和行列式运算

这两种运算都要求所给的矩阵为方阵，在 MATLAB 7.0 语言中，分别使用 inv 函数和 det 函数来实现这两种功能。

- inv 函数可以用来求方阵的逆，*inv*(X) 命令所求值就是方阵 X 的逆。当 X 奇异或范数很小时，系统将给出一个出错信息。
- det 函数可以求方阵的行列式，*det*(X) 命令所求值就是方阵 X 的行列式。

例 9-25 符号矩阵的逆和行列式运算。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> h=sym(hilb(4))
h =
[     1, 1/2, 1/3, 1/4]
[ 1/2, 1/3, 1/4, 1/5]
[ 1/3, 1/4, 1/5, 1/6]
[ 1/4, 1/5, 1/6, 1/7]
>> inv(h)
ans =
[      16,   -120,     240,   -140]
[   -120,    1200, -2700,   1680]
[     240, -2700,    6480, -4200]
[   -140,    1680, -4200,   2800]
>> det(h)
ans =
  1/6048000
>> b=sym('[1,x;x;x^2]')


b =
[1,x],[x],[x^2]
>> inv(b)
??? Error using ==> sym/inv
    Error, invalid terms in product
```

此时，由于 b 是奇异矩阵，系统给出出错警告。

9.3 符号微积分

微积分是大学数学教育的基础，每一个从事科研、教学和学习的人都要经常使用它。MATLAB 7.0 语言提供了许多关于符号微积分计算的功能。

9.3.1　符号极限

极限是微积分的基础和出发点，因此，要想学好微积分，就必须先了解极限的求法，在 MATLAB 7.0 语言中，使用 limit 函数来求符号极限。

- $Limit(F,x,a)$ 命令用来计算符号表达式当 $x \to a$ 时 $F = F(x)$ 的极限值。
- $Limit(F,a)$ 命令用命令 $findsym(x)$ 确定 F 中的自变量，设为变量 x，再计算当 $x \to a$ 时 F 的极限值。
- $Limit(F)$ 命令使用命令 $findsym(x)$ 确定 F 中的自变量，设为变量 x，再计算当 $x \to 0$ 时 F 的极限值。
- $Limit(F,x,a,'right')$ 或 $Limit(F,x,'left')$ 命令用来计算符号函数 F 的单侧极限：左极限 $x \to a-$ 或右极限 $x \to a+$。

例 9-26　使用 limit 函数来求符号极限。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> syms x a t h;
>>  limit(sin(x)/x)
ans =
1
>>  limit((x-2)/(x^2-4),2)
ans =
1/4
>> limit((1+2*t/x)^(3*x),x,inf)
ans =
exp(6*t)
>> limit(1/x,x,0,'right')
ans =
 inf
>> limit(1/x,x,0,'left')
ans =
-inf
>>  limit((sin(x+h)-sin(x))/h,h,0)
ans =
cos(x)
>> limit(v,x,inf,'left')
ans =
[ exp(a),         0]
>>
```

9.3.2　符号微分和求导

在 MATLAB 7.0 语言中，使用 diff 函数来进行微分和求导运算。使用 jacobian 函数实

现对多元符号函数的求导。下面分别予以说明。

1. diff 函数的使用

- $diff(x)$ 命令根据由 $findsym(x)$ 命令返回的自变量 v，求表达式 x 的一阶导数。
- $diff(x,n)$ 命令根据由 $findsym(x)$ 命令返回的自变量 v，求表达式 x 的 n 阶导数，n 必须为自然数。
- $diff(x,'v')$ 或 $diff(S,sym('v'))$ 命令根据由 $findsym(x)$ 命令返回的自变量 v，计算 x 的一阶导数。
- $diff(S,'v',n)$ 命令根据由 $findsym(x)$ 命令返回的自变量 v，计算 x 的 n 阶导数。

例 9-27 使用 diff 函数进行符号微分和求导。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> syms x
>> diff(x^3+3*x^2+2*x+5)
ans =
3*x^2+6*x+2
>> diff(sin(x^3),6)
ans =
-729*sin(x^3)*x^12+7290*cos(x^3)*x^9+17820*sin(x^3)*x^6-9720*cos(x^3)*x^3-360*sin(x^3)
>>
```

以上是求单个自变量时的微分。下面的程序段将对多自变量的函数中的某个变量求导。继续在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> diff(x*y+y^2+sin(x)+cos(y),y)
ans =
x+2*y-sin(y)
>>
>> diff(x*y+y^2+sin(x)+cos(y),y,3)
ans =
sin(y)
>>
```

2 jacobian 函数的使用

$jacobian(f,v)$ 命令用于计算数量或向量 f 对于向量 v 的 Jacobi 矩阵，所得结果的第 i 行第 j 列的数是 $df(i)/dv(j)$。注意当 f 是数量的时候，该命令返回的是 f 的梯度。同时，注意 v 可以是数量，虽然此时 $jacobian(f,v)$ 等价于 $diff(f,v)$。

例 9-28 使用 jacobian 函数求多元函数的导数。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> syms x y z
>> a=[x^2+x*y;sin(x)*cos(y)]
a =
```

```
[          x^2+x*y]
[ sin(x)*cos(y)]
>> jacobian(a,[x,y])
ans =
[      2*x+y,                  x]
[   cos(x)*cos(y), -sin(x)*sin(y)]
>>
```

9.3.3 符号积分

与微分相对应的运算是积分，在 MATLAB 7.0 语言中，使用 int 函数来实现符号积分运算。

- *int*(*S*) 命令根据由 *findsym*(*S*) 命令返回的自变量 *v*，求 *S* 的不定积分，其中 *S* 为符号矩阵或符号数量。如果 *S* 是一个常数，那么积分将针对 *x*。
- *int*(*S*,*v*) 命令对符号表达式 *S* 中指定的符号变量 *v* 计算不定积分。需要注意的是，表达式 *R* 只是函数 *S* 的一个原函数，后面没有带任意常数 *C*。
- *int*(*S*,*a*,*b*) 命令根据由 *findsym*(*S*) 命令返回的自变量 *v*，对符号表达式 S 中的符号变量 *v* 计算从 *a* 到 *b* 的定积分。
- *int*(*S*,*v*,*a*,*b*) 命令对表达式 *S* 中指定的符号变量 *v* 计算从 *a* 到 *b* 的定积分。

例 9-29　用 int 函数求符号积分。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> syms x x1 alpha u t;
>> A = [cos(x*t),sin(x*t);-sin(x*t),cos(x*t)]
A =
[   cos(x*t),   sin(x*t)]
[ -sin(x*t),   cos(x*t)]
>> int(1/(1+x^2))
ans =
atan(x)
>> int(sin(alpha*u),alpha)
ans =
-1/u*cos(alpha*u)
>> int(besselj(1,x),x)
ans =
-besselj(0,x)
>> int(x1*log(1+x1),0,1)
ans =
1/4
>> int(4*x*t,x,2,sin(t))
ans
2*t*(sin(t)^2-4)
```

```
>> int([exp(t),exp(alpha*t)])
ans =
[                    exp(t), 1/alpha*exp(alpha*t)]
>>  int(A,t)
ans =
[ 1/x*sin(x*t),   -cos(x*t)/x]
[     cos(x*t)/x, 1/x*sin(x*t)]
>>
```

9.4 符号积分变换

在科学计算和各种工程实际中，常常要用到各种积分变换，比较常见的有 Fourier 变换及其逆变换、Laplace 变换及其逆变换以及 Z 变换及其逆变换。下面将分别对它们予以介绍。

9.4.1 Fourier 变换及其逆变换

1. Fourier 变换

在 MATLAB 7.0 语言中，使用 fourier 函数来实现 Fourier 变换。其使用格式如下。

- $F = fourier(f)$ 命令将以 x 为默认独立变量，返回符号数量 f 的 Fourier 变换。默认的返回值是关于 w 的一个函数。如果 $f = f(w)$，那么该命令返回一个关于 t 的函数 $F = F(t)$。
- $F = fourier(f,v)$ 命令将返回一个函数 F，该函数以符号 v 为自变量，代替默认符号 w： $fourier(f,v) <=> F(v) = int(f(x)*exp(-i*v*u),x,-inf,inf)$。
- $fourier(f,u,v)$ 命令将返回函数 f，该函数以符号 u 为自变量，代替默认值 x：$fourier(f,u,v) <=> F(v) = int(f(x)*exp(-i*v*u),x,-inf,inf)$。

例 9-30　Fourier 变换的实现。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> syms t v w x
>> fourier(1/t)
ans =
i*pi*(Heaviside(-w)-Heaviside(w))
>> fourier(exp(-x^2),x,t)
ans =
pi^(1/2)*exp(-1/4*t^2)
>> fourier(exp(-t)*sym('Heaviside(t)'),v)
ans =
1/(1+i*v)
```

```
>> fourier(diff(sym('F(x)')),x,w)
ans =
i*w*fourier(F(x),x,w)
>>
```

2. Fourier 变换的逆变换

在 MATLAB 7.0 语言中，使用 ifourier 函数来实现 Fourier 变换的逆变换。其使用格式如下。

- $f = ifourier(F)$ 命令将以 w 为默认独立变量，返回符号数量 F 的 Fourier 变换的逆变换。默认的返回函数是以 x 为自变量的函数：$F = F(w) => f = f(x)$。如果 $F = F(x)$，那么该命令将返回一个关于 t 的函数 $f = f(t)$。
- $f = ifourier(F,u)$ 命令将返回一个函数 f，该函数以符号 u 为自变量，代替默认符号 x：$ifourier(F,u) <=> f(u) = \frac{1}{2pi} * \text{int}(F(w) * \exp(-i * w * u, w, -\text{inf}, \text{inf})$。
- $f = ifourier(F,v,u)$ 命令将返回函数 f，该函数以符号 v 为自变量，代替默认值 w：$ifourier(F,u,v) <=> f(u) = \frac{1}{2pi} \text{int}(F(v) * \exp(-i * u * v), x, -\text{inf}, \text{inf})$。

例 9-31　Fourier 变换的逆变换。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> syms t u v w x
>> ifourier(w*exp(-3*w)*sym('Heaviside(w)'))
ans =
1/2/(-3+i*x)^2/pi
>> ifourier(1/(1 + w^2),u)
ans =
1/2*exp(-u)*Heaviside(u)+1/2*exp(u)*Heaviside(-u)
>> ifourier(v/(1 + w^2),v,u)
ans =
-i/(1+w^2)*Dirac(1,u)
>> ifourier(sym('fourier(f(x),x,w)'),w,x)
ans =
f(x)
>>
```

9.4.2　Laplace 变换及其逆变换

1 Laplace 变换

在 MATLAB 7.0 语言中，使用 laplace 函数实现 Laplace 变换。其使用格式如下。

- $L = laplace(F)$ 命令返回数量符号 F 的以 t 为独立自变量的 Laplace 变换 L。默认

的返回值是一个关于s的函数。如果$F = F(s)$，那么该命令将返回一个关于t的函数$L = L(t)$。

- $L = laplace(F,t)$ 命令返回的函数是一个关于t的函数L，而不是默认的s：$laplace(F,t) <=> L(t) = \text{int}(F(x)*\exp(-t*x),0,\text{inf})$。
- $L = laplace(F,w,z)$ 命令返回的函数L是一个关于z的函数，而不是默认的s：$L = laplace(F,w,z) <=> L(z) = \text{int}(F(w)*\exp(-z*w),0,\text{inf})$。

例 9-32　Laplace 变换。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> syms a s t w x
>> laplace(t^5)
ans =
120/s^6
>> laplace(exp(a*s))
ans =
1/(t-a)
>> laplace(sin(w*x),t)
ans =
w/(t^2+w^2)
>>   laplace(cos(x*w),w,t)
ans =
t/(t^2+x^2)
>> laplace(x^sym(3/2),t)
ans =
3/4/t^(5/2)*pi^(1/2)
>> laplace(diff(sym('F(t)')))
ans =
s*laplace(F(t),t,s)-F(0)
>>
```

2. Laplace 逆变换

在 MATLAB 7.0 语言中，使用 ilaplace 函数实现 Laplace 逆变换

- $F = ilaplace(L)$ 命令返回数量符号L的以t为独立自变量的 Laplace 逆变换F。默认的返回值是一个关于s的函数。如果$L = L(t)$，那么该命令将返回一个关于x的函数$F = F(x)$。
- $F = ilaplace(L,y)$ 命令返回的函数是一个关于y的函数F，而不是默认的t：$ilaplace(L,y) <=> F(y) = \text{int}(L(y)*\exp(s*y),s,c-i*\text{inf},c+i*\text{inf})$。
- $F = ilaplace(L,y,x)$ 命令返回的函数F是一个关于x的函数，而不是默认的t：$ilaplace(L,y,x) <=> F(y) = \text{int}(L(y)*\exp(x*y),y,c-i*\text{inf},c+i*\text{inf})$。

例 9-33　Laplace 逆变换。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> syms s t w x y
>> ilaplace(1/(s-1))
ans =
exp(t)
>> ilaplace(1/(t^2+1))
ans =
sin(x)
>> ilaplace(t^(-sym(5/2)),x)
ans =
4/3*x^(3/2)/pi^(1/2)
>> ilaplace(y/(y^2 + w^2),y,x)
ans =
cos((w^2)^(1/2)*x)
>> ilaplace(sym('laplace(F(x),x,s)'),s,x)
ans =
F(x)
>>
```

9.4.3　Z 变换及其反变换

1. Z 变换

在 MATLAB 7.0 语言中，使用 ztrans 函数实现 Z 变换。

- $F = ztrans(f)$ 命令返回数量符号 f 的以 n 为独立自变量的 Z 变换 F。默认的返回值是一个关于 z 的函数：$f = f(n) => F = F(z)$。f 的 Z 变换定义成 $F(z) = symsum(f(n)/z\hat{}n, n, o, \inf)$。如果 $f = f(z)$，那么该命令将返回一个关于 w 的函数 $F = F(w)$。
- $F = ztrans(f, w)$ 命令返回的函数是一个关于 w 的函数 F，而不是默认的 z：$ztrans(f, w) <=> F(w) = symsum(f(n)/w\hat{}n, n, 0, \inf)$。
- $F = ztrans(f, k, w)$ 命令返回函数 f 关于 k 的 Z 变换函数：$ztrans(f, k, w) <=> F(w) = symsum(f(k)/w\hat{}k, k, 0, \inf)$。

例 9-34　Z 变换。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> syms k n w z
>> ztrans(2^n)
ans =
1/2*z/(1/2*z-1)
>> ztrans(sin(k*n),w)
ans =
w*sin(k)/(w^2-2*w*cos(k)+1)
>> ztrans(cos(n*k),k,z)
```

```
ans =
(z-cos(n))*z/(z^2-2*z*cos(n)+1)
>> ztrans(cos(n*k),n,w)
ans
(w-cos(k))*w/(w^2-2*w*cos(k)+1)
>> ztrans(sym('f(n+1)'))
ans =
z*ztrans(f(n),n,z)-f(0)*z
>>
```

2. Z 的逆变换

在 MATLAB 7.0 语言中，使用 iztrans 函数实现 Z 的逆变换。

- $f = iztrans(F)$ 命令返回数量符号 F 的以 z 为独立自变量的 Z 的逆变换 f。默认的返回值是一个关于 n 的函数：$F = F(z) => f = f(n)$。如果 $F = F(n)$，那么该命令将返回一个关于 k 的函数 f = f(k)。
- $f = iztrans(F, k)$ 命令返回的函数是一个关于 k 的函数 f，而不是默认的 n，这里 m 是一个数量符号。
- $f = iztrans(F, w, k)$ 命令将 F 看成是 w 的函数而不是默认的 $sym\,\mathrm{var}(F)$，它返回的函数 f 是关于 k 的 Z 的逆变换函数：$F = F(w)$ 和 $f = f(k)$。

例 9-35　Z 的逆变换。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> syms x z k
>> iztrans(z/(z-2))
ans =
2^n
>> iztrans(exp(x/z),z,k)
ans =
x^k/k!
>>
```

9.5　符号代数方程的求解

本节将介绍如何运用 MATLAB 7.0 语言求解符号代数方程，包括线性方程组的求解和非线性方程组的求解。

9.5.1　符号线性方程组的求解

在 MATLAB 7.0 语言中，使用 linsolve 函数来求解符号代数方程。

使用格式为 $X = linsolve(A, B)$，它的结果与 $X = sym(A)/sym(B)$ 相同。

例 9-36　使用 linsolve 函数求解符号线性方程组。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> a=sym('rand(4)')
a =
rand(4)
>> b=('[1;2,3;4]')
b =
[1;2;3;4]
>> linsolve(a,b)
ans =
[ 1/proc () local t; global _seed, _seed := irem(427419669081* seed,999999999989); t :  _seed;
irem(t,4) end]
[ 2/proc () local t; global _seed; _seed := irem(427419669081*_seed,999999999989); t := _seed,
irem(t,4) end]
[ 3/proc () local t, global _seed; _seed := irem(427419669081*_seed,999999999989), t := _seed;
irem(t,4) end]
[ 4/proc () local t, global _seed; _seed := irem(427419669081*_seed,999999999989), t = _seed;
irem(t,4) end]
>>
```

9.5.2　符号非线性方程组的求解

在 MATLAB 7.0 中，使用 fsolve 函数和求解 S 非线性方程组。

设 $F(X)=0$，其中 F 和 X 可以是向量或矩阵。

- $X = fsolve(fun, X_0)$ 命令以 X_0 为初始矩阵来求解方程 fun，fun 接受输入量 X 并返回一个向量(矩阵)，使得 $F = fun(X)$。
- $X = fsolve(fun, X_0, options)$ 命令以 $options$ 为选择参数的输入变量，详细内容用户可以查看 optimset 的帮助信息得知。
- $X = fsolve(fun, X_0, options, P1, P2)$ 命令将问题定性参数 $P1$、$P2$ 和 $P3$ 等直接赋值给函数 $fun(X, P1, P2, P3...)$。而当 $options$ 为默认值时，该命令将返回一个空矩阵。
- $[X, fval] = fsolve(fun, P1, P2, P3...)$ 命令返回客观方程在 X 处的值。
- $[X, fval, exitflag] = fsolve(fun, X_0...)$ 命令返回一个描述 $fsolve$ 的溢出情况的字符串 $exitflag$。

  当 $exitflag > 0$ 时，$fsolve$ 的解将收敛到 X。

当 $exitflag = 0$ 时，将取得方程的最大解数。

当 $exitflag < 0$ 时，$fsolve$ 的解在 X 处不收敛。

- $[X, fval, exitflag, output, jacob] = fsolve(fun, X_0 ...) ...)$ 命令返回函数 fun 在 X 处的 Jacobian 解。

例 9-37 使用 fsolve 函数求解非线性方程组。

方程组为 $\begin{cases} 3*x(1)^\wedge 2 - x(2)^\wedge 2 = 0 \\ 3*x(1)*x(2)^\wedge 2 - x(1)^\wedge 2 - 1 = 0 \end{cases}$

程序段如下。

```
function y=myfun2(x)
y(1)=3*x(1)^2-x(2)^2;
y(2)=3*x(1)*x(2)^2-x(1)^2-1;
```

以上程序为 M 文件中的程序段，继续在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> x=[0.8 0.4]
x =
     0.8000      0.4000
>> x = fsolve('myfun2',x)
x =
     0.5208      0.9020
>>
```

9.5.3 一般符号代数方程组的求解

在 MATLAB 7.0 语言中，使用 solve 函数求解一般的符号代数方程组。其使用格式如下。

- $solve('eqn1','eqn2',...,'eqnN')$
- $solve('eqn1','eqn2',...,'eqnN','var1, var2,..., var N')$
- $solve('eqn1','eqn2',...,'eqnN','var1', var2',...,'var N')$

上边各式中的 *eqns* 是一些具体方程的符号表达式或字符串。而 *vars* 是一些未知的符号变量或未知的字符串。

sowe 将找出这些表达式的 0 点或这些方程的解。

例 9-38 使用 solve 函数求解一般代数方程组。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>>   solve('p*sin(x) = r')
ans =
asin(r/p)
>>
```

这里 x 是未知量，继续在命令窗口中输入以下程序，并按 Enter 键确认。

```
>> [x,y] = solve('x^2 + x*y + y = 3','x^2 - 4*x + 3 = 0')
x =
[ 1]
[ 3]
y =
[    1]
[ -3/2]
>>
```

继续在命令窗口中输入以下程序，并按 Enter 键确认。

```
>>S = solve('x^2*y^2 - 2*x - 1 = 0','x^2 - y^2 - 1 = 0')
S =
     x: [8x1 sym]
     y: [8x1 sym]
>>
```

这里 S 是一个结构体。继续在命令窗口中输入以下程序，并按 Enter 键确认。

```
>> [u,v] = solve('a*u^2 + v^2 = 0','u - v = 1')
u =
[ 1/2/(a+1)*(-2*a+2*(-a)^(1/2))+1]
[ 1/2/(a+1)*(-2*a-2*(-a)^(1/2))+1]
v =
[ 1/2/(a+1)*(-2*a+2*(-a)^(1/2))]
[ 1/2/(a+1)*(-2*a-2*(-a)^(1/2))]
>>
```

这里 a 被作为参数求解关于 u 和 v 的方程组。继续在命令窗口中输入以下程序，并按 Enter 键确认。

```
>>S = solve('a*u^2 + v^2','u - v = 1','a,u')
S =
     a: [1x1 sym]
     u: [1x1 sym]
>>
```

这里 v 被作为参数来求解方程组，并且返回 S.a 和 S.u。继续在命令窗口中输入以下程序，并按 Enter 键确认。

```
>> [a,u,v] = solve('a*u^2 + v^2','u - v = 1','a^2 - 5*a + 6')
a =
[ 2]
[ 2]
[ 3]
```

```
[ 3]
u =
[ 1/3+1/3*i*2^(1/2)]
[ 1/3-1/3*i*2^(1/2)]
[ 1/4+1/4*i*3^(1/2)]
[ 1/4-1/4*i*3^(1/2)]
v =
 [ -2/3+1/3*i*2^(1/2)]
[ -2/3-1/3*i*2^(1/2)]
[ -3/4+1/4*i*3^(1/2)]
[ -3/4-1/4*i*3^(1/2)]
>>
```

上边的程序段求解了关于 a、u 和 v 的方程组。继续在命令窗口中输入以下程序，并按 Enter 键确认。

```
>> [x,y] = solve('sin(x+y)-exp(x)*y = 0','x^2-y = 2')
x =
-6.0173272500593065641097297117905
y =
34.208227234306296508646214438330
>>
```

由于系统无法给出符号解，因此系统自动给出数值解。

9.6 符号微分方程的求解

在 MATLAB 7.0 语言中，使用 dsolve 函数求解常微分方程。其使用格式为 *dsolve('eqn1','eqn2',...)*。

其中，对给定的常微分方程(组)eq1、eq2……中指定的符号自变量 v，与给定的边界条件和初始条件 cond1、cond2……。求符号解(即解析解)r：若没有指定变量 v，则默认变量为 t；在微分方程(组)的表达式 eq 中，大写字母 D 表示对自变量(设为 x)的微分算子：$D=d/dx$、$D2=d^2/dx^2$……。微分算子 D 后面的字母则表示为因变量，即待求解的未知函数。初始和边界条件由字符串表示，如 $y(a)=b$、$Dy(c)=d$ 和 $D2y(e)=f$ 等，分别表示 $y(x)|_{x=a}=b$，$y'(x)|_{x=c}=d$，$y''(x)|_{x=e}=f$；若边界条件少于方程(组)的阶数，则返回的结果 r 中会出现任意常数 C1、C2……；dsolve 命令最多可以接受 12 个输入参量(包括方程组与定解条件个数，当然我们可以做到输入的方程个数多于 12 个，只要将多个方程置于一字符串内即可)。若没有给定输出参量，则在命令窗口显示解列表。若该命令找不到解析解，则返回警告信息，同时返回一个空的 sym 对象。这时，用户可以用命令 ode23 或

ode45 求解方程组的数值解。

例 9-39　使用 dsolve 函数求解符号微分方程组。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> dsolve('Dx = -a*x')
ans =
C1*exp(-a*t)
>> x = dsolve('Dx = -a*x','x(0) = 1','s')
x =
exp(-a*s)
>> y = dsolve('(Dy)^2 + y^2 = 1','y(0)   0')
y =
[   sin(t)]
[ -sin(t)]
>> S = dsolve('Df = f + g','Dg = -f + g','f(0) = 1','g(0) = 2')
S =
      f. [1x1 sym]
      g. [1x1 sym]
>> Y = dsolve('Dy = y^2*(1-y)')
Warning: Explicit solution could not be found; implicit solution returned.
> In C:\MATLAB 7.06p5\toolbox\symbolic\dsolve.m at line 292
 Y =
t+1/y-log(y)+log(-1+y)+C1=0
>> dsolve('Df = f + sin(t)', 'f(pi/2) = 0')
ans =
-1/2*cos(t)-1/2*sin(t)+1/2*exp(t)/(cosh(1/2*pi)+sinh(1/2*pi))
>> dsolve('D2y = -a^2*y', 'y(0) = 1, Dy(pi/a) = 0')
ans =
cos(a*t)
>> S = dsolve('Dx = y', 'Dy = -x', 'x(0)=0', 'y(0)=1')
S =
      x: [1x1 sym]
      y: [1x1 sym]
>> S = dsolve('Du=v, Dv=w, Dw=-u','u(0)=0, v(0)=0, w(0)=1')
S =
      u: [1x1 sym]
      v: [1x1 sym]
      w: [1x1 sym]
>> w = dsolve('D3w = -w','w(0)=1, Dw(0)=0, D2w(0)=0')
w =
1/3*exp(-t)+2/3*exp(1/2*t)*cos(1/2*t*3^(1/2))
>>
```

9.7 图示化符号函数计算器

与其他的高级语言相比，MATLAB 7.0 语言的一个重要优点是简单易学，在符号运算方面，MATLAB 7.0 同样体现了这个特点。MATLAB 7.0 语言提供了图示化符号函数计算器，用户可以进行一些简单的符号运算和图形处理。虽然它的功能不是十分强大，但是，由于它操作方便，使用简单，可视性和人机交互性都很强，因此深得用户喜欢。MATLAB 7.0 语言有两种符号函数计算器，一种是单变量符号函数计算器；另一种是泰勒级数逼进计算器。

9.7.1 单变量符号函数计算器

在 MATLAB 7.0 语言中，使用 funtool 函数来调用图示化单变量符号函数计算器。用户在命令窗口中直接输入 funtool 命令，即可将图示化符单变量符号函数计算器调出，如图 9-3 所示。下面介绍一下它的功能和用法。

funtool 命令将生成 3 个图形窗口，Figure No.1 用于显示函数 f 的图形(显示图形窗口 1)，Figure No.2 用于显示函数 g 的图形(显示图形窗口 2)，Figure No.3 为一可视化的、可操作与显示一元函数的计算器界面(控制窗口)。在该界面上有许多按钮，可以显示两个由用户输入的函数的计算结果：加、乘、微分等。funtool 还有一个函数存储器，允许用户将函数存入，以便后面调用。在开始时，funtool 显示两个函数 $f(x) = x$ 与 $g(x) = 1$ 在区间[-2* π，2* π]上的图形。Funtool 同时在下面显示一控制面板，允许用户对函数 f、g 进行保存、更正、重新输入、联合与转换等操作。注意在任何情况下，这 3 个图形中只能有一个处于激活状态。如图 9-3 中 Figure No.3 处于激活状态，而 Figure No.1 和 Figure No.2 处于非激活状态。

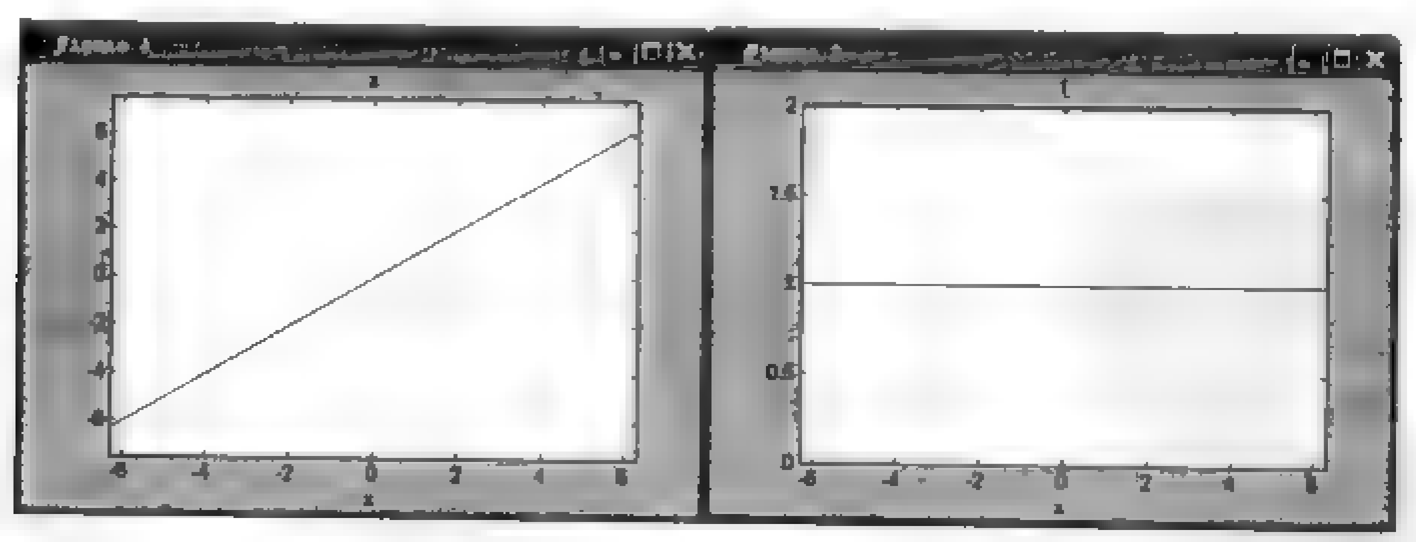


图 9-3 图示化单变量符号函数计算器

图 9-3　(续)

1. 输入框的功能

如图 9-3 所示，控制窗口中一共有 4 个文本框，分别是“f =”、“g =”、“x =”和“a =”。用户使用图示化符号函数计算器，就是在这 4 个窗口中输入相关的数据来进行操作。下面介绍一下控制窗口中这 4 个文本框的功能。

- “f =”文本框显示代表函数 f 的符号表达式，它的默认值是 x，用户可以在该行输入其他有效的表达式来定义 f，再按 Enter 键即可在显示图形窗口 1 中绘制出图形。
- “g =”文本框显示代表函数 g 的符号表达式，它的默认值是 1，用户可以在该行输入其他有效的表达式来定义 g，再按 Enter 键即可在显示图形窗口 2 中绘制出 g 的图形。
- “x =”文本框显示用于函数 f 与 g 的绘制区间，它的默认值为[-2π<s<2π,-2π<t<2π]。用户可以在该行输入其他的不同区间，再按 Enter 键即可改变显示图形窗口 1 与显示图形窗口 2 中的区间。
- “a =”文本框显示一个用于改变函数 f 的常量因子(见下面的操作按钮)，它的默认值为 1/2。用户可以在该行输入不同的常数。

2. 控制按钮的功能

在文本框的下边，是 4 行控制按钮，用户可以使用它们来进行对函数的一些运算操作，并取得一些帮助信息。

(1) 运算操作按钮的功能

前 3 行操作按钮用于对在文本框中输入的函数进行各种操作，其中第 1 行的按钮用于函数自身的操作，第 2 行的按钮用于函数 f 和常数 a 之间的操作，第 3 行的按钮用于函数 f 和函数 g 之间的操作。它们的使用功能如下。

- df/dx：函数 f 的导数；
- int f：函数 f 的积分(没有常数的一个原函数)，当函数 f 的原函数不能用初等函数表示时，操作可能失败；
- simple f：化简函数 f(若有可能)；
- num f：函数 f 的分子；
- den f：函数 f 的分母；

- 1/f：函数 f 的倒数；
- finv：函数 f 的反函数，若函数 f 的反函数不存在，操作可能失败；
- f+a：用 f(x)+a 代替函数 f(x)；
- f-a：用 f(x)-a 代替函数 f(x)；
- f*a：用 f(x)+a 代替函数 f(x)；
- f/a：用 f(x)/a 代替函数 f(x)；
- f^a：用 f(x)^a 代替函数 f(x)；
- f(x+a)：用 f(x+a)代替函数 f(x)；
- f(x*a)：用 f(x-a)代替函数 f(x)；
- f+g：用 f(x)+g(x)代替函数 f(x)；
- f-g：用 f(x)-g(x)代替函数 f(x)；
- f*g：用 f(x)*g(x)代替函数 f(x)；
- f/g：用 f(x)/g(x)代替函数 f(x)；
- g=f：用函数 f(x)代替函数 g(x)；
- swap：函数 f(x)与 g(x)互换；

(2) 系统操作按钮的功能

第 4 行的操作按钮用于控制进行函数计算的各种操作，并提供各种在线帮助信息。它们的使用功能如下。

- Insert：将函数 f(x)保存到函数内存列表中的最后；
- Cycle：用内存函数列表中的第二项代替函数 f(x)；
- Delete：从内存函数列表中删除函数 f(x)；
- Reset：重新设置计算器为初始状态；
- Help：显示在线的关于计算器的帮助；
- Demo：运行该计算器的演示程序；
- Close：关闭计算器的三个窗口。

例 9-40　单变量符号函数计算器。

在"g ="文本框中输入 x^3，并连续 3 次单击 Cycle 操作按钮之后，再单击 swap 操作按钮，所得结果如图 9-4 所示。

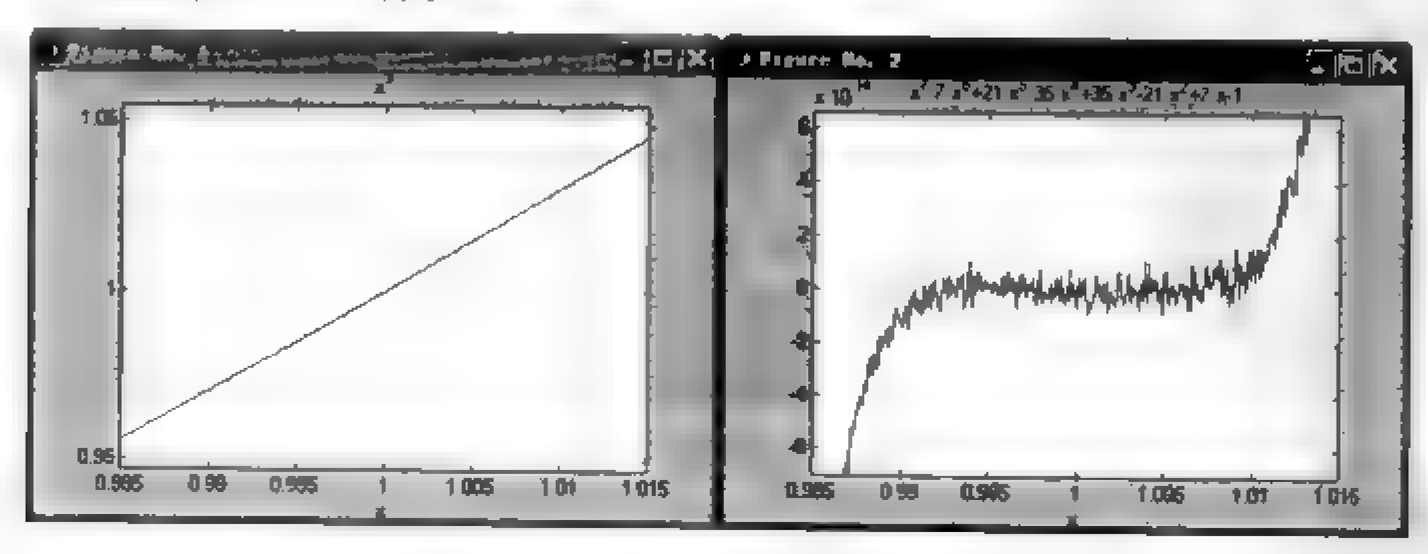


图 9-4　单变量符号函数计算器

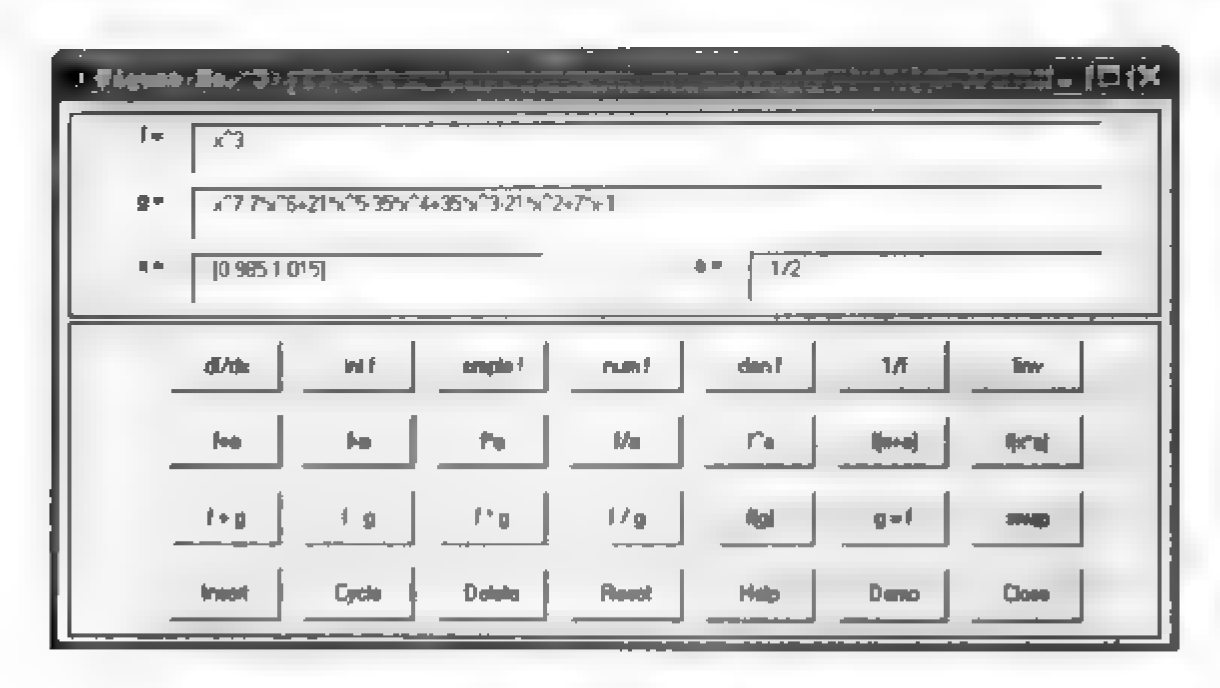


图 9-4 (续)

9.7.2 泰勒级数逼近计算器

在 MATLAB 7.0 语言中，使用 taylortool 函数来调用图示化泰勒级数逼近计算器。用户在命令窗口中直接输入 taylortool 命令，即可将图示化泰勒级数逼近计算器调出。下面介绍一下它的功能和用法。

taylortool 是一个交互式的泰勒级数逼近计算器，它可以绘制函数 f 前 N 阶的泰勒级数。默认区间是[-2*π,2*π]，默认的阶数为 7 阶，默认的函数为 *f=x*cos(x)*，如图 9-5 所示。

taylortool(f) 在[-2*π2*π]区间内绘制函数 *f* 从第 1 阶到第 N 阶的部分泰勒级数和。默认的 N 值为 7。

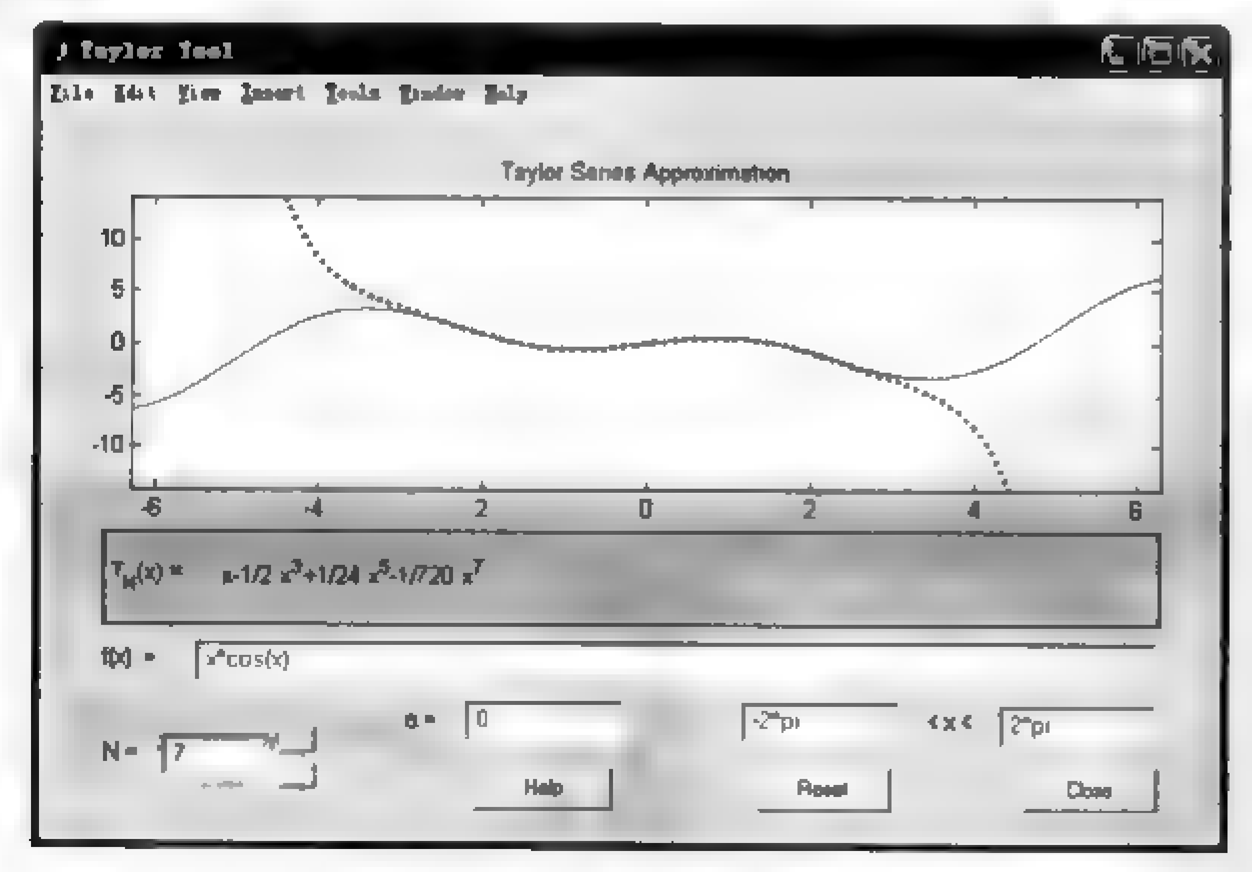


图 9-5 泰勒级数逼近计算器

例 9-41 求函数函数 $f(x)=\sin(x)*\cos(x)$ 在区间[-π, π]的 10 阶泰勒级数。

用户在“f(x)=”文本框中输入“sin(x)*cos(x)”，在“N=”文本框中输入“10”，在“<x<”文本框的左右两边输入“-π”和“π”。按Enter键确认后，即得如图9-6所示得泰勒级数逼进图。

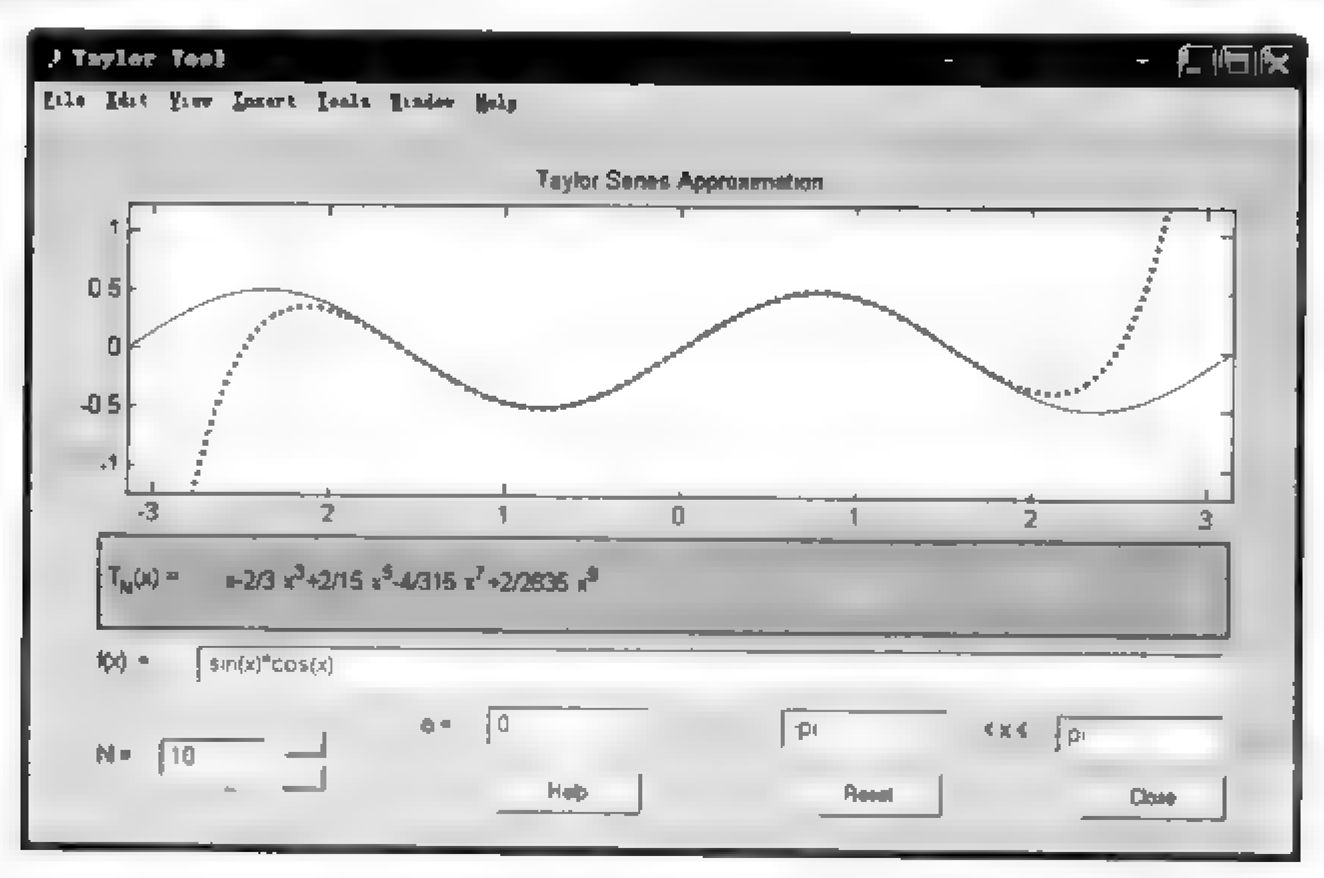


图9-6　$f(x)=sin(x)*cos(x)$在区间[-π, π]的10阶泰勒级数

9.8　利用Maple的深层符号计算资源

尽管MATLAB 7.0已经提供了十分强大的符号计算功能，但是作为一门优秀的语言，它的另一个成功之处在于可以和其他的语言实现良好的兼容。Maple语言提供了2000多条符号函数计算命令，当用户需要进行非常复杂的符号运算时，就可以通过MATLAB 7.0直接调用Maple中的命令。在MATLAB 7.0语言中，用户可以使用maple和mfun两个函数来直接调用Maple中的函数。

9.8.1　maple命令的调用

maple是Symbolic Toolbox工具箱的一个通用命令，使用它可以实现对Maple中大部分函数的调用。其使用格式如下。

- *maple*(*statement*)命令将*statement*传递给Maple软件，其中*statement*是一个合乎Maple语法的可执行语句的字符串。为了满足Maple语言的语法，必要的时候可以在*statement*里边添加分号。该命令的输出结果也符合Maple语法。
- *maple*('*function*', *ARH*1, *ARG*2,...,)命令接受所调用的任何函数及其相关参数，其中*function*为所输入的函数名，*ARG*1和*ARG*2等为该函数的参数，Maple语法的执行语句由*function*(*Arg*1, *ARG*2,...,)构成，也就是说，在输入语句直接由

逗号隔开，所有的输入语句必须是符合 Maple 语法的字符串，其输出的结果也是符合 Maple 语法的字符串。可以使用 sym 函数将一个 Maple 字符串转换为 Symbolic 下的有效语句。

例 9-42　求递推方程 $f(n)=4*f(n-1)-3*f(n-2)$ 的通解。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> ty1=maple('rsolve(f(n)=4*f(n-1)-3*f(n-2),f(k));')
ty1 =
-(1/2*f(0)-1/2*f(1))*3^k+3/2*f(0)-1/2*f(1)
>>
```

以上是调用 Maple 函数的第一种格式，用户也可以使用另外一种格式，继续在命令窗口中输入如下程序，并按 Enter 键确认。

```
>> ty2=maple('rsolve','f(n)=-4*f(n-1)-3*f(n-2)','f(k)')
ty2 =
(3/2*f(0)+1/2*f(1))*(-1)^k+(-1/2*f(0)-1/2*f(1))*(-3)^k
>>
```

例 9-43　求 $f = xyz$ 的 Hessian 矩阵。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> af1=maple('hessian(x*y*z,[x,y,z]);')
af1 =
matrix([[0, z, y], [z, 0, x], [y, x, 0]])
>>
```

以上是调用 Maple 函数的第一种格式，用户也可以使用另外一种格式，继续在命令窗口中输入如下程序，并按 Enter 键确认。

```
>> af2=maple('hessian','x*y*z','[x,y,z]')
af2 =
matrix([[0, z, y], [z, 0, x], [y, x, 0]])
>>
```

用户还可以把以上的输出变为符号类，使用 sym 函数来实现这项功能。以上边已经生成的程序段为例，我们可以进行如下操作。

```
>>af=sym(af2)
af =
[ 0, z, y]
[ z, 0, x]
[ y, x, 0]
>>
```

此时，af2 已经转换成为符号类。

9.8.2 mfun 命令的使用

mfun 函数用于给 Maple 函数赋值，其使用格式如下。

mfun('fun', p1, p2,...pk) 命令中，*fun* 是 Maple 函数的函数名，而 *p1*、*p2* 和 *pk* 等为 Maple 函数的输入参数。最后一个参数可以为矩阵，其他所有的参数必须和 Maple 函数的规定类型相吻合。*mfun* 函数使用给定的参数求出 *fun* 函数的值并以双精度的数值形式返回 MATLAB 7.0。对于函数 *fun* 的任何不连续，系统都将返回 NaN。

例 9-44 mfun 函数的使用。

解：在命令窗口中输入如下命令，并按 Enter 键确认，结果如果 9-7 所示。

```
>> x = 0:0.1:5.0;
>> y = mfun('FresnelC',x);
>> plot(x,y)
>>
```

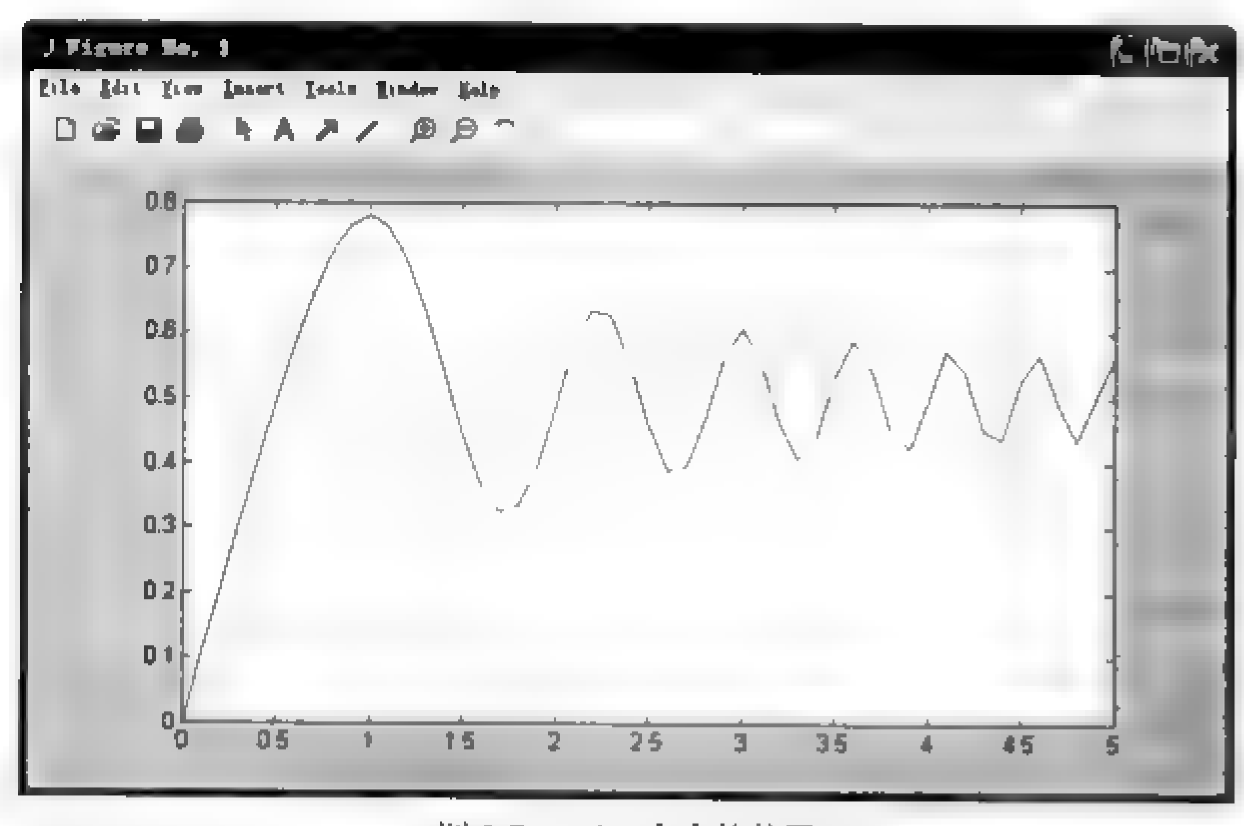


图 9-7 mfun 命令的使用

9.8.3 maple 库函数在线帮助的检索树

为了获取关于 maple 库函数的帮助信息，用户可以用 mhelp index、mhelp index[category] 和 mhelp fun_name 命令来获得各种类型的帮助信息。

1. mhelp index 命令

该命令可以翻阅 Maple 在线帮助的索引类目录。使用格式如下。

```
mhelp index        Index of help descriptions
Calling Sequence:
        ?index[category]    or    help(index, category);
```

Description.
- The following categories of topics are available in the help subsystem:
index[expression]##expression　operators for forming expressions
index[function]##function　list of Maple functions
index[misc]##misc　miscellaneous facilities
index[package]##packages　descriptions of library packages
index[procedure]##procedure topics related to procedures and programming
index[statement]##statement　list of Maple statements
To access these help pages, you must prefix the category with index, thus ? index[category].

2. mhelp index[category]命令

该命令可以深入 Maple 的具体分类目录，其使用格式如下。

mhelp index[function]
Index of descriptions for standard library functions
Description:
- The following are the names of Maple's standard library functions. For more information, see ?f where f is any of these functions.

AFactor##AFactor　AFactors##Afactors
AiryZeros##AiriAiZeros　AiryZeros##AiriBiYZeros
Airy##AiryAi　Airy##AiryBi
AngerJ##AngerJ　Berlekamp##Berlekamp
Bessel##BesselI　Bessel##BesselJ
...
...
hattype##whattype　with##with
worksheet##worksheet　writebytes##writebytes
writedata##writedata　writeline##writeline
writestat##writestat　writeto##writeto
zip##zip　ztrans##ztrans
See Also.
readlib##readlib, libname##libname, index[package]##index[package]

3. 获取具体函数使用方法说明的指令格式是 mhelp fun_name

```
>> mhelp fresnel
FresnelC - The Fresnel Cosine Integral
FresnelS - The Fresnel Sine Integral
Fresnelf, Fresnelg - The Fresnel Auxiliary Functions
Calling Sequence
        FresnelC(x)
        FresnelS(x)
        Fresnelg(x)
        Fresnelf(x)
Parameters
```

```
        x - an expression
    Description:
    - The Fresnel cosine integral is defined as follows.
          FresnelC(x) =    int(cos(Pi/2*t^2), t=0..x);
    - The Fresnel sine integral is defined as follows:
          FresnelS(x) =    int(sin(Pi/2*t^2), t=0..x);
    - The Fresnel auxiliary functions are defined as follows:
          Fresnelf(x) = (1/2 - FresnelS(x))*cos(Pi/2*x^2) -
                                            (1/2 - FresnelC(x))*sin(Pi/2*x^2)
          Fresnelg(x) = (1/2 - FresnelC(x))*cos(Pi/2*x^2) -
                                            (1/2 - FresnelS(x))*sin(Pi/2*x^2)
    Examples:
    > FresnelS(infinity);
                                                1/2
    > FresnelC(1);
                                             FresnelC(1)
    > evalf(%);
                                             .7798934004
    > Fresnelf(1.0);
                                             .2798934004
    See Also:
    erf, dawson, inifcns
    >>
```

9.9 习　题

1. 创建符号表达式 $f(x)=\sin(x)+\cos(x)-\tan(x)$ 。

2. 计算习题 9.1 中的符号表达式在 $x=0$ 、$x=1$ ，$x=2\pi$ 处的值。

3. 计算符号表达式 $f(x)=x+\cos(x)-\sin(x)$ 在 $x=\pi-1$ 处的值，并将结果设置为以下 5 种精度，即分别为小数点之后 1 位、2 位、10 位、20 位和 50 位有效数字。

4. 设 x 为符号变量，$f(x)=x^4+2x^2+1$ ，$g(x)=x^3+6x^2+3x+5$ ，试进行如下运算。

(1) $f(x)+g(x)$

(2) $f(x)\times g(x)$

(3) 对 $f(x)$ 进行因式分解

(4) 求 $g(x)$ 的反函数

5. 设 x 为符号变量，令 $y=x^4+2x^2+1$ ，$z=y^2$ ，$w=\sin(z)$ ，试求 w 关于 x 的符号表达式，并求当 $x=5$ 时， w 的小数点之后具有 1 位、2 位、10 位、20 位和 50 位有效

数字的数值解。

6. 使用 sym 函数生成如下符号矩阵，$a = sym('[1/x,1/(x+1);1/(x+2),1/(x+3)]')$，$b = sym('[x,1;x+2,0]')$，试对这两个符号矩阵分别进行如下操作。

(1) $a-b$

(2) $a*b$

(3) 求 a 的倒置

(4) 求 a 的行列式

(5) 求 b 的逆

(6) 求 a 的秩

(7) 求 a^3

7. 求 $\frac{\tan(x)}{x}$ 当 $x \to 0$ 时的极限。

8. 求 $\frac{\sin(x)}{x-\pi}$ 当 $x \to \pi$ 时的极限。

9. 求 $\sin(x)+x$ 在 $x=[0,8]$ 上的定积分。

10. 求符号表达式 $\sin(x)+x^5$ 的 5 次微分。

11. 对符号表达式 $z = x.*e^{-x.^2-y.^2}$，分别进行如下变换。

(1) 关于 x 的 Fourier 变换及其逆变换

(2) 关于 y 的 Laplace 变换及其逆变换

(3) 分别关于 x 和 y 的 Z 变换及其逆变换

12. 求解线性方程组 $\begin{cases} 3x+4y=1 \\ 4x+3y=-1 \end{cases}$

13. 求解非线性方程组 $\begin{cases} x-0.7\sin(x)-0.2\cos(y)=0 \\ y-0.7\cos(x)+0.2\sin(y)=0 \end{cases}$

第10章　MATLAB 7.0程序设计

通过前面几章的学习，用户可以体会到 MATLAB 语言与其他语言相比的巨大优势，用户可以在命令窗口中直接输入命令行，从而以一种交互式的方式来编写程序。这种方式适用于命令行比较简单，输入比较方便，同时处理的问题步骤较少的情况。当需要处理重复、复杂且容易出错的问题时，单纯使用这种方式就会比较吃力。因此，作为一门高级语言，MATLAB 7.0 和其他高级语言(例如 C、Fortran 和 Basic 等)一样，可以进行控制流的程序设计，这就是 M 文件的编程工作方式。

10.1　M 文件入门

10.1.1　M 文件的基本特点

M 文件的语法类似于一般高级语言，是一种程序化的编程语言，但是，与传统的高级语言相比，M 文件又有自己的特点。它只是一个简单的 ASCII 型码文本文件，因此，它的语法比一般的高级语言要简单，程序也容易调试，并且有很好的交互性。

从语言特点上来说，MATLAB 7.0 是一种解释性的语言，它本身不能做任何事情，而只是对用户发出的指令起解释执行的作用。因此，MATLAB 7.0 在初次运行 M 文件时会将 M 文件编成代码并装入内存中，此过程会大大降低程序的运行速度，但是，用户在再次运行该程序时，系统将直接从内存中取出代码运行，此时程序的运行速度将极大加快。

MATLAB 语言提供了很多的工具箱，工具箱中的函数就是一个个的 M 文件，正是有了这些工具箱，MATLAB 7.0 才可以广泛地应用到各个领域，如动态仿真、CDMA 参数模块集、通信模块集、通信工具箱、控制系统工具箱和数字信号工具箱等。根据需要，用户可以在这些工具箱中添加自己的 M 文件，注意每个 M 文件必须以 m 为扩展名。

由于 MATLAB 7.0 语言是由 C 语言编写的，因此，它的语法与 C 语言有很大的相似之处，对于熟悉 C 语言或是对 C 语言有初步了解的用户来说，学习 MATLAB 7.0 将是一件十分简单的事情。

M 文件有两种，一种为脚本式(Script)，一种为函数式(Function)。它们各有自己的特点，下面将予以介绍。

1. M 文件的基本属性

函数 M 文件必须满足一些标准，另外，它们还应该满足一些 MATLAB 7.0 的属性。主要有以下几点。

(1) 函数式 M 文件名和出现在文件的第一行的函数名必须相同。实际上，MATLAB 7.0 忽略了第一行的函数名，并且根据存储在硬盘上的文件名来执行函数。

(2) 函数的文件名最多可以有 31 个字符。这个最大值由操作系统限制决定，有的系统允许的最大字符数会更少。MATLAB 7.0 忽略第 31 个字符以为的或者超出了操作系统限制决定的字符以为的字符，因此可以使用稍长一些的名字，这样就可以为每一个 M 文件提供一个惟一的文件名。

(3) 函数式 M 文件名在 Unix 平台上对大小写是敏感的，但是在 Windows 平台上是不分大小写的。为了实现在不同平台中的通用性，建议对 M 文件只使用小写。

(4) 函数名必须以一个字母开头，开头之后，可以是任意的字母、数字和下划线的组合。这个命名规则与变量的命名规则相同。

(5) 一个函数式 M 文件的第一行被称为“函数声明行”，而且函数式 M 文件必须包括 function 这个词，其后就是这个函数最常用的方式调用的语法。在第一行声明的输入和输出变量是这个函数的局部变量。输入变量包含传递给这个函数的数据，输出变量包含从这个函数输出的变量。通过输入变量将数据输出来是不行的。

(6) 在函数声明行之后的第一个连续的注释行的集合是这个函数的帮助文本。第一个注释行被称作 H1 行，这一行也是 lookfor 命令搜索的行。H1 行通常包括大写的函数名以及这个函数功能的一个简要描述。在第一行之后的注释行描述了可能的调用语法所使用的算法，而且可能会有简单的示例。

(7) 在一个函数的帮助文本中出现的函数名通常都是大写的，这样做的目的就是让函数名在视觉上突出，而函数调用的时候都是用小写字母。

(8) 在第一个连续注释行集合之后的所有语句构成了函数体。一个函数的函数体包含了对输入参数进行运算并且将运算结果赋值给输出参数的 MATLAB 7.0 语句。

(9) 一个函数式 M 文件在这个文件的最后一行被执行完毕或者在任何地方遇到 return 语句的时候就终止。

(10) 一个函数可以通过调用函数 error 来异常终止操作指令运行并将控制权返还给命令窗口。

(11) 通过调用函数 warning，一个函数可以发出一条警报信息然后继续执行。这个函数对于报告意外的或者式其他一些异常行为来说是非常有用的。

(12) 函数式 M 文件可以包含对脚本文件的调用。当遇到一个脚本文件时，这个脚本文件就在这个函数的工作区执行，而不是在 MATLAB 7.0 的工作区间执行。

(13) 在一个函数式 M 文件中可以出现多个函数。这些函数被称作子函数或是局部函数，子函数以一个标准的函数声明语句开始，并且遵循所有的函数创建规则。

(14) 子函数可以被这个 M 文件中的子函数调用，也可以被这个 M 文件中的其他函数调用。同所有的函数一样，子函数也有自己独立的工作区间。

(15) 除了子函数之外，M 文件还可以调用私有的 M 文件，这种私有的 M 文件是驻留在标识为 private 的调用函数的子目录下的标准 M 文件。只有在私有 M 文件的父目录下的函数才能访问这些私有 M 文件。

2. M 文件的组成部分

下面介绍一下 M 文件的基本组成部分。以一个简单的例子予以说明。

例 10-1　函数式 M 文件的组成部分。

解：调出 Medit 窗口，在里边输入如下内容。

```
function f = fact(n)                              %函数的定义行
% Compute a factorial value.                      %H1 行
% FACT(N) returns the factorial of N,             %帮助文本
% usually denoted by N!
% Put simply, FACT(N) is PROD(1:N).               %注释
f = prod(1:n);                                    % 函数体
```

表 10-1 列出了 M 文件各个部分的功能和作用。

表 10-1　M 文件的组成部分及其功能

组 成 部 分	描　　述
函数定义行(仅限于函数式 M 文件)	定义函数名，以及输入和输出变量的数目和顺序
H1 行	H1 行对程序进行概括性的描述，使用 help 和 lookfor 命令都可以调出此行
帮助文本	这是比 H1 行更详细的帮助信息，使用 help 命令时与 H1 行一起显示
函数体	函数体是 M 文件的主要部分，程序的计算和设计都在此实现
注释	解释程序行的意义

10.1.2　脚本式 M 文件

有时候用户需要输入较多的命令，而且经常要对这些命令进行重复输入，此时，直接在命令窗口输入显得比较麻烦，而利用命令文件就显得比较方便和简单。用户可以将需要重复输入的所有命令按顺序放到一个扩展名为 m 的文本文件下，每次运行时只要输入该 M 文件的文件名即可。需要注意的是，用户自己创建的 M 文件的文件名要避免与 MATLAB 7.0 的内置函数和工具箱中的函数重名，以免发生内置函数被替换的情况。同时，当用户所创建的 M 文件不在当前搜索路径时，该函数将无法调用。

由于脚本式文件的运行相当于在命令窗口中顺次输入运行命令，在编制这类文件时，只需将所要执行的语句逐行编辑到指定的文件中，且变量不需要预先定义，在命令文件中的变量都是全局变量，任何其他的命令文件和函数都可以访问这些变量，也不存在文件名对应的问题。

- 在命令窗口中直接输入 edit 命令，或是单击常用工具栏上的“新建”图标，可以打开一个新的 M 文件编辑窗口。
- 如果用户要编辑某个已经存在的 M 文件，可以使用 edit mfiles 命令的形式，其中 mfiles 为用户需要编辑的文件名。

◆ 运行 M 文件时，一定要保证所调用的 M 文件在当前的路径下，否则，MATLAB 7.0 将无法找到需要调用的函数，从而给出错误信息。用户可以使用 which 函数来证实所调用函数是否在当前路径下。

例如，要查找 plot 函数是否在当前路径下时，可以在命令窗口输入 which plot，并按 Enter 键确认。

```
>> which plot
C:\MATLAB 7.07\toolbox\MATLAB 7.0\graph2d\plot.bi
>>
```

如果所用命令不在当前路径下时，可以使用函数 addpath 来设置路径。例如，可以使用命令 addpath('C: MATLAB 7.07\toolbox')将路径设置在 C:\MATLAB 7.07\toolbox 目录下。

例 10-2　编一个命令文件，求 sin(1)、sin(2)……sin(10)的值。

解：在编辑窗口中输入语句如下。

```
%这是一个关于脚本式的 M 文件的例子
%主要用于介绍脚本式 M 文件的生成
%该函数用于顺次求出从 sin(1)到 sin(10)的值。
for i=1:10
      a=sin(i);
   fprintf('sin(%d)=',i)
      fprintf('%12.8f\n',a)
end
```

将该 M 文件以文件名 sumsin.m 保存在 work 文件夹下面，如图 10-1 所示。在命令窗口中输入 sumsin，并按 Enter 键确认，得到以下输出结果。

```
>> sumsin
sin(1)=   0.84147098
sin(2)=   0.90929743
sin(3)=   0.14112001
sin(4)=  -0.75680250
sin(5)=  -0.95892427
sin(6)=  -0.27941550
sin(7)=   0.65698660
sin(8)=   0.98935825
sin(9)=   0.41211849
sin(10)= -0.54402111
>>
```

可见，使用脚本式的 M 文件，可以使程序的输入变得简单、快速和方便。我们也可以查看 sumsin.m 文件的帮助信息，在命令窗口中输入 help sumsin 即可得到关于 sumsin 脚本式 M 文件的相关信息。

```
>> help sumsin
 这是一个关于脚本式的 M 文件的例子
 主要用于介绍脚本式 M 文件的生成
 该函数用于顺次求出从 sin(1)到 sin(10)的值。
>>
```

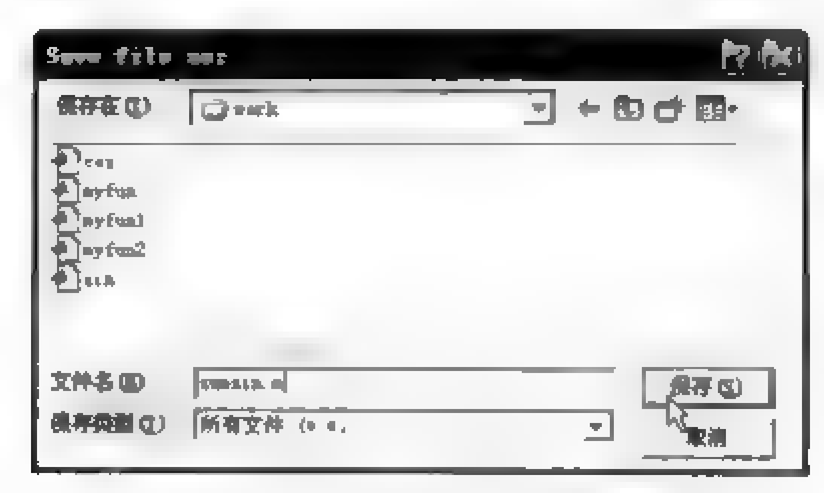


图 10-1　脚本式 M 文件的保存

例 10-3　编辑一个脚本式 M 文件绘制一个正弦曲线。

解：调出 Medit 窗口，在里边输入如下内容。

```
%此文件用于绘制【−2×pi，2×pi】区间的正弦曲线图
x=-2*pi:0.05:2*pi;
y=sin(x);
plot(x,y,'c+')
legend('正弦曲线图')
```

将该文件以 sinpic.m 为文件名保存在 work 文件夹下边。在命令窗口中输入 sinpic，并按 Enter 键确认，得到以下输出结果，如图 10-2 所示。

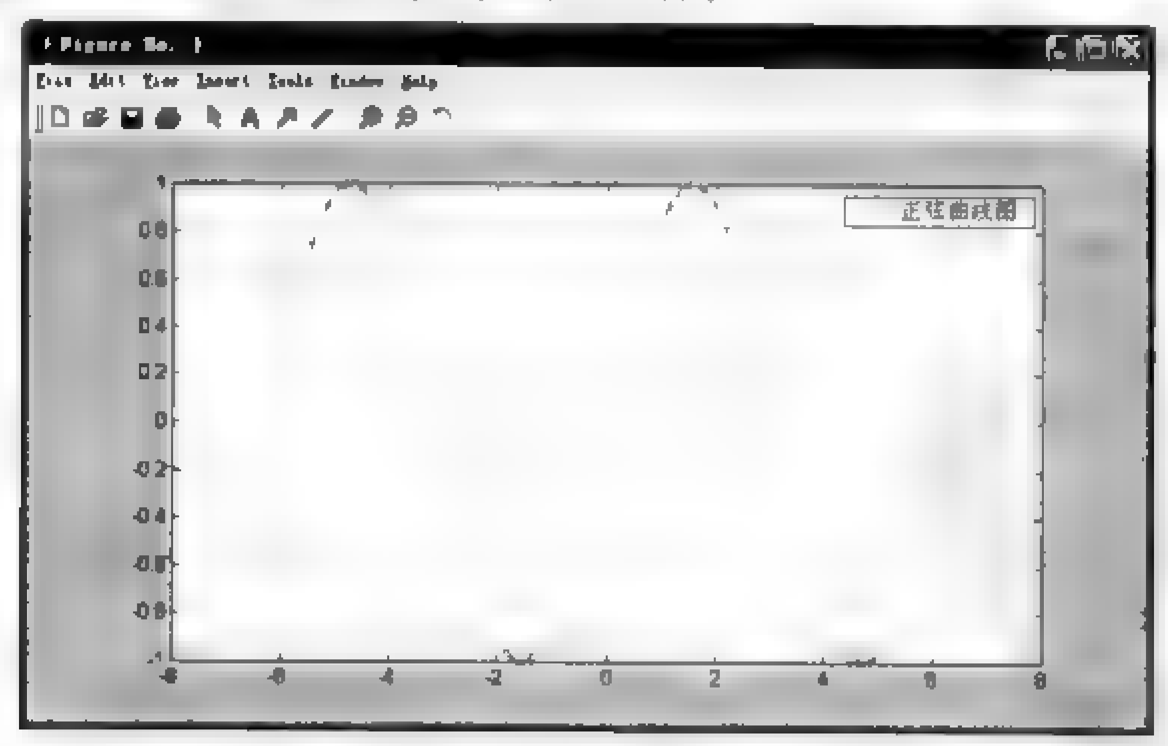


图 10-2　使用脚本式 M 文件绘制图形

10.1.3　函数式 M 文件

函数式 M 文件比脚本式 M 文件相对要复杂一些，脚本式 M 文件只是将一些命令语句组织在一起，不需要自带参数，也不一定要返回结果，而函数式 M 文件一般都要自带参数，并且有返回结果，这样，就可以更好地把整个程序连为一段。当然，也有一些函数式 M 文件不带参数，这时，文件中一般使用一些全局变量来实现与外界和其他函数之间的数据交换。

函数式 M 文件的第一行都是以 function 开始，说明此文件是一个函数。其实质就是用户往 MATLAB 7.0 函数库里边添加的子函数。函数式 M 文件中的变量都不是全局变量，仅在函数运行期间有效，函数运行完毕之后，它所定义的变量将从工作区间中清除。

下面的几个例子将说明函数式 M 文件的使用方法。

例 10-4　判断某一年是否为闰年。

解：调出 Medit 窗口，在里边输入如下内容。

```
%该函数用于判断某一年是否为闰年
%使用格式为 isleapea(y),其中 y 是用户所要计算的年份
function isleapyear(year)
sign=0;
if rem(year,4)==0
    sign=sign+1;
end
if rem(year,100)==0
          sign=sign-1;
 end
 if rem(year,400)==0
          sign=sign+1;
  end
  if sign==1
        fprintf('%4d year is a leap year.\n',year)
  else
        fprintf('%4d year is not a leap year.\n',year)
  end
```

将该文件以 isleapyear.m 为文件名保存在 work 文件夹中。用户可以使用如下方法调用 isleapyear 函数。

```
>>y1=1000;
>> y2=2000;
>> y3=1996;
>> y4=3000;
>> isleapyear(y1)
1000 year is not a leap year.
```

```
>> isleapyear(y2)
2000 year is a leap year.
>> isleapyear(y3)
1996 year is a leap year.
>> isleapyear(y4)
3000 year is not a leap year.
>>
```

下面再列举一个例子，说明比较脚本式 M 文件和函数式 M 文件中变量的作用范围的区别。

例 10-5　脚本式 M 文件和函数式 M 文件中变量的作用范围的区别。

解：首先创建函数式 M 文件如下。

```
%该程序用于检验函数式 M 文件中变量的存储方式
%即检验它的变量是否为全局变量或是局部变量
function y=hanshu(a)
x1=a*3
x2=3/a
x3=3+a
x4=3-a
```

将函数命名为 hanshu.m，并在命令窗口中给 x1、x2、x3 和 x4 赋值如下。

```
>> x1=0.1;
>> x2=0.2;
>> x3=0.3;
>> x4=0.4;
```

此时，赋值已经完成，给定一个 a 值，运行程序如下。

```
>> a=2;
>>hanshu(a)
x1 =
      6
x2 =
    1.5000
x3 =
      5
x4 =
      1
```

此时，在函数式 M 文件内部，x1、x2、x3 和 x4 的值已经计算出来了，与前边的赋值相比有了很大的变化。但是，在工作区间内，x1、x2、x3 和 x4 的值并没有发生变化，如下面程序所示。

```
>> x1
```

```
x1 =
     0.1000
>> x2
x2 =
     0.2000
>> x3
x3 =
     0.3000
>> x4
x4 =
     0.4000
>> whos
  Name       Size                    Bytes  Class
  a          1x1                         8  double array
  x1         1x1                         8  double array
  x2         1x1                         8  double array
  x3         1x1                         8  double array
  x4         1x1                         8  double array
Grand total is 5 elements using 40 bytes
>>
```

以上程序充分说明，在函数式 M 文件中，变量的值是以局部变量的形式存储在文件中的，这与脚本式 M 文件截然不同。

把上面的函数式 M 文件修改为脚本式 M 文件如下。

```
%该程序用于检验脚本式M文件中变量的存储方式
%即检验它的变量是否为全局变量或是局部变量
x1=a*3
x2=3 a
x3=3+a
x4=3-a
```

将函数命名为 mingling.m.，并仍在命令窗口中给 x1、x2、x3 和 x4 赋值如下。

```
>> x1=0.1,
>> x2=0.2;
>> x3=0.3;
>> x4 0.4;
```

此时，赋值已经完成，给定一个 a 值，运行程序如下。

```
>> a=2;
>> mingling
x1 =
      6
x2 =
```

```
      1.5000
x3 =
       5
x4 =
       1
```

此时，在脚本式 M 文件内部，x1、x2、x3 和 x4 的值已经计算出来了，与前边的赋值相比有了很大的变化。但是，目前还不知道在工作区间内，x1、x2、x3 和 x4 的值有没有发生变化，用户可以通过如下程序检验。

```
>> x1
x1 =
       6
>> x2
x2 =
      1.5000
>> x3
x3 =
       [illegible]
>> x4
x4 =
       1
>>
>> whos
  Name        Size                  Bytes  Class
   a           1x1                      8  double array
   x1          1x1                      8  double array
   x2          1x1                      8  double array
   x3          1x1                      8  double array
   x4          1x1                      8  double array
Grand total is 5 elements using 40 bytes
>>
```

可见，使用脚本式 M 文件时，在 M 文件中改变的值将带到工作区间来，用户在使用时，一定要注意这两种不同文件在这一点上的区别。

10.2　MATLAB 7.0 程序控制

要想编写好的 MATLAB 7.0 程序，就有必要学好 MATLAB 7.0 语言的控制语句，在 MATLAB 7.0 语言中，程序的控制极其重要，用户只有熟练掌握了这方面的内容，才能编制出高质量的程序。最简单的程序控制就是顺序结构，用户依次输入命令语句即可。此外，MATLAB 7.0 语言还提供了 4 种高级的控制结构。它们是：if-else-end

结构、switch-case-otherwise-end 结构、for 循环和 while 循环。由于这些结构经常包含大量的 MATLAB 7.0 命令，因此经常出现在 M 文件中，而不是直接加在 MATLAB 7.0 提示符下。

10.2.1　顺序结构

顺序结构是最简单的程序结构，用户在编写好程序之后，系统将按照程序的物理位置顺次执行。因此，这种程序比较容易编制。但是，由于它不包含其他的控制语句加子结构等内容，程序结构比较单一，实现的功能也比较有限。尽管如此，对于比较简单的程序来说，使用顺序结构还是能够很好地解决。

例 10-6　顺序结构的实现。

解：编写 M 文件如下。

```
a=1;
b=2;
c=3;
s1=a+b
s2=s1+c
s3=s2/s2
```

将该文件以文件名 shunxujieguo.m 保存在 work 文件夹中，在命令窗口中输入 shunxujieguo，并按 Enter 键确认，得到如下结果。

```
>> shunxujieguo
s1 =
     3
s2 =
     6
s3 =
     1
>>
```

可见，系统依次执行各条命令语句，并将结果显示出来。

10.2.2　选择语句

在编写程序时，往往需要根据一定的条件，进行一定的选择来执行不同的语句，此时，需要使用分支语句来控制程序的进程。在 MATLAB 7.0 语言中，使用 if-else-end 结构来实现这种控制。

if-else-end 结构的使用形式有以下 3 种。

1. 只有一种选择时的情况

此时的程序结构如下。

```
if 表达式
    执行语句
end
```

这是该结构最简单的一种应用形式，它只有一个判断语句，当表达式为真时，就执行 if 和 end 语言之间的执行语句；否则不予执行。

例 10-7 只有一种选择的情况下 if-else-end 选择语句的使用。

解：编制 M 文件如下。

```
%该函数用于演示 if-else-end 语句的第一种用法
function f=ifone(x)
if x>0
    fprintf('%f is a positive number\n',x)
end
```

在上边的程序中，当参数 x 为正数时，系统将执行语句 fprintf('%f is a positive number\n',x)，否则系统将不作任何反应。用户可以验证如下。

```
>> x=4;
>> ifone(x)
4.000000 is a positive number
>>
```

由于 x>0，所以执行了 fprintf('%f is a positive number\n',x)，但是当 x<0 或 x=0 时，运行程序的结果如下。

```
>> x1=-9;
>> ifone(x1)
>>x2=0;
>>ifone(x2)
>>
```

可见，系统没有执行任何操作。

2. 有两种选择时的情况

假如有两个选择，if-else-end 结构是如下。

```
if 表达式
   执行语句 1
else
    执行语句 2
end
```

此时，如果表达式为真，则系统将运行执行语句 1；如果表达式是假，则系统将运行执行语句 2。

例 10-8　有两种选择的情况下 if-else-end 选择语句的使用。

解：编制 M 文件如下。

```
%该程序用于演示有 2 种选择时 if-else-end 语句的使用
function iftwo(x)
if x>0
    fprintf('%f is a positive number\n',x)
else
    fprintf('%f is not a positive number\n',x)
end
```

在上边的程序中，当参数 x 为正数时，系统将执行语句 fprintf('%f is a positive number\n',x)，否则系统将执行语句 fprintf('%f is not a positive number\n',x)。用户可以验证如下。

```
>> x1=5.3;
>> iftwo(x1)
5.300000 is a positive number
>> x2=0;
>> iftwo(x2)
0.000000 is not a positive number
>> x3=-9.234;
>> iftwo(x3)
-9.234000 is not a positive number
>>
```

可见，此时的程序可以根据输入的数据进行不同的操作，来实现不同的功能。

3. 有 3 种或 3 种以上选择时的情况

当有 3 种或更多的选择时，if-else-end 结构采用形式如下。

```
if 表达式 1
    表达式 1 为真时的执行语句 1
  elseif 表达式 2
        表达式 2 为真时的执行语句 2
  elseif 表达式 3
        表达式 3 为真时的执行语句 3
  elseif 表达式 4
        表达式 4 为真时的执行语句 4
  elseif……
           ……
           ……
```

```
        ……
    else
        所有的表达式都为假时的执行语句
    end
```

在这种形式中，当运行到程序的某一条表达式为真时，则执行与之相关的执行语句，此时系统将不再检验其他的关系表达式，即系统将跳过其余的 if-else-end 结构。而且，最后的 else 命令可有可无。

例 10-9　有 3 种或 3 种以上选择的情况下 if-else-end 选择语句的使用。

解：编写 M 文件如下。

```
%该函数用于演示 If—else—end 语句的第 3 种用法
function f=ifthree(x)
if x>100
    fprintf('%f is a   great positive number\n',x)
  elseif x>=10
        fprintf('%f is a   big positive number\n',x)
  elseif x>10
        fprintf('%f is a   small positive number\n',x)
elseif x==0
        fprintf('%f is zero\n',x)
  else
        fprintf('%f is minus number\n',x)
end
```

在上面的程序中，共有 5 种分支，根据不同的参数，程序会做出不同的响应。如当 x>100 时，程序将执行语句 fprintf('%f is a great positive number\n',x)。

```
>> x1=120;
>> x2=23;
>> x3=5;
>> x4=0;
>> x5=-7.8;
>> ifthree(x1)
120.000000 is a   great positive number
>> ifthree(x2)
23.000000 is a   big positive number
>> ifthree(x3)
5.000000 is minus number
>> ifthree(x4)
0.000000 is zero
>> ifthree(x5)
-7.800000 is minus number
>>
```

从上面的程序可以看出，使用 if-else-end 结构可以很方便地解决具有多个分支的复杂问题。

10.2.3　分支语句

在 MATLAB 7.0 语言中，除了上边介绍的 if-else-end 分支语句外，还提供了另外一种分支语句形式，那就是 switch - case - otherwise - end 分支语句，这是 MATLAB 6.0 以上的版本新增加的功能。这可以使熟悉 C 语言或者其他高级语言的用户更方便地使用 MATLAB 7.0 的分支功能。它的使用格式如下。

```
switch 开关语句
      case 条件语句，
        执行语句，……，执行语句
      case {条件语句 1，条件语句 2，  条件语句 3，……}
         执行语句，……，执行语句
      ……
      otherwise,
         执行语句，……，执行语句
   end
```

在上边的分支结构中，当某个条件语句的内容与开关语句的内容相匹配时，系统将执行其后的语句，如果所有的条件语句与开关条件都不相符合时，系统将执行 otherwise 后边的语句。

例 10-10　switch - case - otherwise - end 分支语句的使用。

解：编制 M 文件如下。

```
function lower1(method)
switch method
          case {'linear','bilinear'}
            disp('Method is linear')
          case 'cubic'
            disp('Method is cubic')
          case 'nearest'
            disp('Method is nearest')
          otherwise
            disp('Unknown method.')
        end
```

将上边的程序以 lower1.m 为文件名保存，在命令窗口中运行该文件。

```
>> a='linear';
>> lower1(a)
Method is linear
```

```
>> a='bilinear';
>> lower1(a)
Method is linear
>> b='cubic';
>> lower1(b)
Method is cubic
>> c='My God';
>> lower1(c)
Unknown method.
>>
```

可见，程序成功地反映了作者的意图，可以很方便地实现不同的分支选择。

10.2.4 模块

try-catch 模块给用户提供了一种错误捕获机制。换句话说，利用 try-catch 模块，MATLAB 7.0 编译系统发现的错误将被其捕获，用户可以控制 MATLAB 7.0 怎样对发生的错误进行处理。它的使用格式如下。

```
try
   执行语句 1
catch
   执行语句 2
end
```

一般来说，执行语句 1 中的所有命令都要执行。如果执行语句 1 中没有 MATLAB 7.0 错误出现，那么，在执行完执行语句 1 之后，出现控制就直接跳到 end 语句；但是，如果在运行执行语句 1 的过程中，出现了 MATLAB 7.0 错误，那么，程序控制就立即转移到 catch 语句，然后执行执行语句 2。在 catch 模块中，函数 lasterr 包含了在 try 模块中遇到的错误生成的字符串。这样，catch 模块中的执行语句 2 就可以获取这个错误字符串，然后采取相应的动作。

try-catch 模块也可以嵌套使用，即在一个 try-catch 模块的执行语句 1 和执行语句 2 中都可以嵌入子 try-catch 模块。

例 10-11　try-catch 模块的应用举例。

解：在命令窗口中输入如下语句，并按 Enter 键确认。

```
>> X=magic(4);
>> Y=ones(4,3);
>> try
        Z=X*Y
catch
      Z=nan;
```

```
    disp('X and Y is not conformable.')
end

Z=

     34     34     34
     34     34     34
     34     34     34
     34     34     34

>>
```

在上边的程序中，由于 X 和 Y 的维数兼容，所以执行语句“Z=X*Y”没有语法错误，程序得以执行，故程序跳过 catch 下边的执行语句“Z=nan;”和“disp('X and Y is not conformable.')”。如果改变矩阵 Y 的维数，使得矩阵 X 和 Y 不能相乘，MATLAB 7.0 程序将运行执行语句 2。继续在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> X=magic(4);
>> Y=ones(3,3);
>> try
   Z=X*Y
catch
disp('Xand Y is not conformable.')
end
X and Y is not conformable.
>>
```

此时，使用 lasterr 函数可以调出所犯的错误。

```
>> lasterr
ans =
Error using ==> mtimes
Inner matrix dimensions must agree.
>>
```

10.2.5　for 循环语句

前边介绍了两种重要的分支结构，使用这两种结构，用户可以很方便地控制程序的进程，从而使程序结构清晰，便于操作。但是，当遇到许多有规律的重复运算时，单纯使用前边所学的知识就比较困难了。同其他的高级语言一样，MATLAB 7.0 语言也提供了循环语句，可以让用户很方地实现循环操作，从而可以从容地应付大规模的循环语句。MATLAB 7.0 语言提供了两种循环方式，即 for 循环和 while 循环，本节将具体介绍 for 循环。

(1) for 循环的最大特点是，它的循环判断条件通常就是对循环次数的判断，也就是说，

循环语句的循环次数是预先设定好的。它的使用格式如下。

```
for i =表达式,
    执行语句, ……, 执行语句
end
```

它的表达式是一个向量，其形式可以是 *m:s:n*；其中 *m*、*s* 和 *n* 可以为整数、小数或是负数。但是当 *n>m* 时，*s* 必须为大于 0 的数，而当 *n<m* 时，*s* 必须为小于 0 的数。因为只有这样，表达式才能组成一个向量。表达式也可以为 *m:n* 这样的形式，此时，*s* 的值默认为 1，*n* 必须大于 *m*。用户还可以直接将一个向量赋值给 i，此时程序将穷尽该向量的每一个值。i 还可以是字符串、字符串矩阵或由字符串组成的单元阵。

例 10-12　for 循环的使用。

解：在命令窗口中输入如下语句，并按 Enter 键确认。

```
>> for i=1:1:10
        x(i)=i^2;
end
>> x
x =
     1     4     9    16    25    36    49    64    81   100
>>
```

该程序使用一个 for 循环，求出了数组 x 从 x(1)到 x(10)的值，程序运行完之后，自动在工作区间生成了双精度变量 i 和双精度数组 x。用户可以通过使用 whos 命令来查看相关的情况。

```
>> whos
  Name      Size                    Bytes  Class

  i         1x1                         8  double array
  x         1x10                       80  double array

Grand total is 11 elements using 88 bytes

>>
```

(2) for 循环的另一个特点是嵌套使用，它可以多次嵌套 for 循环或是和其他的结构形式嵌套使用，这样用户就可以利用它实现更为复杂的功能。下面就是一个嵌套使用的简单实例。

例 10-13　for 循环的嵌套使用。

解：在命令窗口中输入如下语句，并按 Enter 键确认。

```
>> for i=1:5,
for j=1:5,
```

```
a(i,j)=1/(i+j-1),
end
end
>> a
a =
    1.0000    0.5000    0.3333    0.2500    0.2000
    0.5000    0.3333    0.2500    0.2000    0.1667
    0.3333    0.2500    0.2000    0.1667    0.1429
    0.2500    0.2000    0.1667    0.1429    0.1250
    0.2000    0.1667    0.1429    0.1250    0.1111
>>
```

例 10-14　使用 for 循环求 $\sum_{i=1}^{10} i!$ 以及 I! (I=1:10)的值。

解：打开 M 文件编辑窗口，输入程序如下，并将函数命名为 forsum。

```
sum=0;
for i=1:10,
    part=1,
    for j=1:i,
        part=part*j,
    end
    fprintf('part(%d)=%d.\n',i,part);
    sum=sum+part;
end
fprintf('The total sum is %d.\n',sum);
```

在命令窗口中输入 forsum，即得到如下结果。

```
>> forsum
part(1)=1.
part(2)=2.
part(3)=6.
part(4)=24.
part(5)=120.
part(6)=720.
part(7)=5040.
part(8)=40320.
part(9)=362880.
part(10)=3628800.
The total sum is 4037913.
>>
```

10.2.6　while 循环语句

与 for 循环不同，while 循环的判断控制可以是逻辑判断语句，因此，它的循环次数可以是一个不定数。这样就赋予了它比 for 循环更广泛的用途。它的使用格式如下。

```
while 表达式
    执行语句
end
```

在这个循环中，只要表达式的值不为 false，程序就会一直运行下去，用户必须注意的是，当程序设计出了问题，比如表达式的值总是 true 时，程序容易陷入死循环。因此在使用 while 循环时一定要在执行语句中设置使表达式的值为 false 的情况。

例 10-15　while 循环的使用。

解：在命令窗口中输入如下语句，并按 Enter 键确认。

```
 >> A=rand(3)
A =
     0.9501     0.4860     0.4565
     0.2311     0.8913     0.0185
     0.6068     0.7621     0.8214
>> E = 0*A;
F = E + eye(size(E)); N = 1;
            while norm(E+F-E,1) > 0,
               E = E + F;
               F = A*F/N;
               N = N + 1;
            end
>> F
F =
   1.0e-016 *
     0.1658     0.2159     0.1001
     0.0574     0.0747     0.0346
     0.1813     0.2361     0.1094
>> E
E =
     3.1250     1.7453     1.1993
     0.6360     2.6358     0.1796
     1.8010     2.2981     2.6663
>> N
N =
     23
>>
```

10.2.7　人机交互命令

前边已经介绍了关于 MATLAB 7.0 语言的基本控制语句，用户可以使用它们进行比较复杂的程序设计。此外，MATLAB 7.0 还提供了一些特殊的程序控制语句，用户可以使用它们提前终止循环、跳出子程序以及显示 M 文件的执行过程等操作。从而使得用户在程序设计时能够与计算机进行及时的交互，使得程序设计变得更为得心应手，所设计的程序也能变得更加合理。

1. 终止命令 break 和 return

首先介绍 break 命令，break 语句一般用在循环控制中，如 for 循环和 while 循环，一般通过 if 语句来使用 break 语句，当 if 语句满足一定的条件时，break 语句将被调用，系统将在循环尚未结束时跳出当前循环。在多层嵌套循环中，break 只能跳出包含它的最内层的循环。

例 10-16　break 语句的使用。

解：编制 M 文件如下。

```
%该程序用于求解经典的鸡兔同笼问题。
%鸡兔同笼，头 36，脚 100。
%求鸡兔各几只。
i=1;
while i>0
    if rem(100-i*2,4)==0&(i+(100-i*2)/4)==36
        break;
    end
    i=i+1;
    n1=i;
    n2=(100-2*i)/4;
    break
end
    fprintf('The number of chicken is %d.\n',n1);
    fprintf('The number of rabbit is %d.\n',n2);
```

将该 M 文件以 chicken 为文件名保存，并在命令窗口中输入 chicken，运行该程序得到如下结果。

```
>> chicken
The number of chicken is 2.
The number of rabbit is 24
>>
```

但是，由于 break 语句只应用于循环语句中，当用户要在循环语句以外进行终止操作时，可以使用 return 命令，执行 return 命令后，进程将返回调用函数或是键盘，同时，使

用 return 命令可以终止 keyboard 模式。通常，程序在 end 处结束，而 return 命令可以提前结束程序。

例 10-17　return 语句的使用方法。

解：编制 M 文件如下。

```
function d = det(A)
if isempty(A)
    d = 1;
    return
else
    .
end
```

2. 继续命令 continue

continue 语句一般也是用在循环控制中，如 for 循环和 while 循环，一般通过 if 语句来使用 continue 语句，当 if 语句满足一定的条件时，continue 语句将被调用。与 break 语句不同的是，使用 continue 语句之后，系统只是不再执行相关的执行语句，而不会跳出当前循环。

例 10-18　continue 语句的使用。

解：编制 M 文件如下。

```
for i=1:10
    for j=1:10
     if mod(j,2)==1
     continue
    else
        fprintf('%d        ',i+j)
    end
  end
  fprintf('\n');
 end
```

将上边的程序以文件名 continue2.m 保存，在命令窗口中输入 continue2，并按 Enter 键确认，得到如下结果。

```
>> continue2
3       5       7       9       11
4       6       8       10      12
5       7       9       11      13
6       8       10      12      14
7       9       11      13      15
8       10      12      14      16
9       11      13      15      17
10      12      14      16      18
11      13      15      17      19
```

```
    12    14    16    18    20
>>
```

3. 等待用户反应命令 pause

pause 命令用于使程序暂时终止运行，等待用户按任意键后继续运行。这非常适合于用户在调试程序时需要查看中间结果的情况。它的使用格式如下。

- pause 命令用于暂时终止程序，按 Enter 键后继续。
- pause (n)命令在程序运行到该处时将等待 n 秒后继续。这里 n 可以是分数，对于大部分计算机来说，n 精确到 0.01 秒是没有问题的。
- pause off 命令意味着随后的 pause 和 pause (n)命令将不予执行，这使得交互式的文本能够不被干扰地执行完。
- pause on 命令意味着随后的 pause 和 pause (n)命令将予以执行。

例 10-19　pause 命令的使用介绍。

解：编制 M 文件如下。

```
%该程序用于显示 PAUSE 函数的使用方法
%程序一开始绘制出一条正弦曲线，然后进入 pause 状态
%当用户按 Enter 键后，系统将绘制出一条余弦曲线，然后又进入 pause(n)状态
%等待 10 秒后，系统将绘制出一条正弦和余弦的和的曲线
 x=0:0.05:6*pi;
 y=sin(x);
 z=cos(x);
 r=y+z;
plot(x,y);
 pause
 plot(x,z)
 pause(10)
 plot(x,r)
```

将该程序以 pause1.m 为文件名保存，在命令窗口中输入 pause1，并按 Enter 键确认，一次得到如下结果。

```
>> pause1
>>
```

按 Enter 键确认后，系统将绘制出一条正弦曲线，如图 10-3 所示，并进入 pause 状态。等待用户按 Enter 键确认。

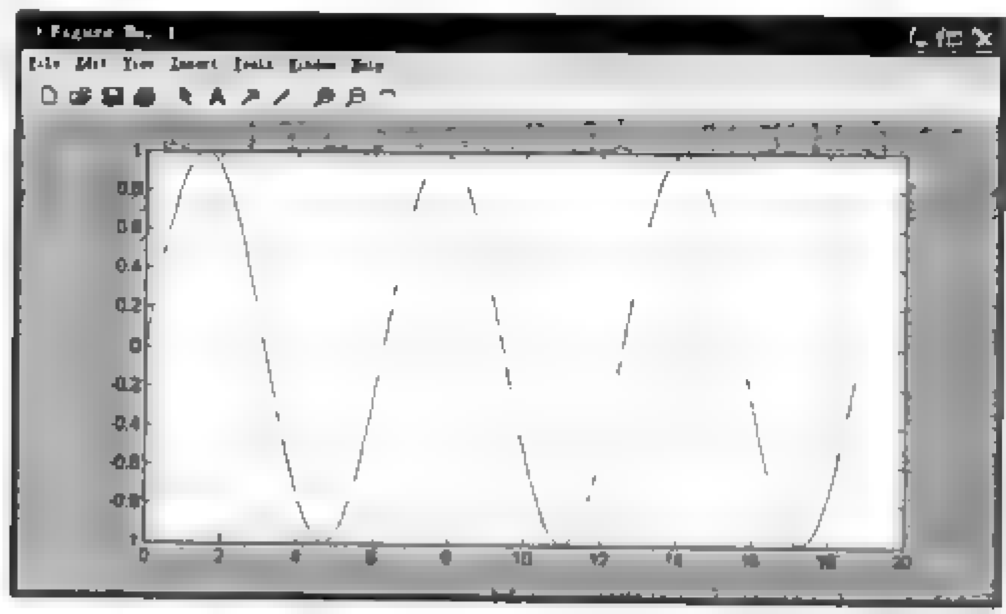


图 10-3　pause 命令的使用介绍 1

用户按 Enter 键后，系统将依照程序进程，绘制一条余弦曲线，如图 10-4 所示。此后系统进入 pause(n)状态，这里 n=10，即 10 秒后，系统将自动执行下一步的操作。

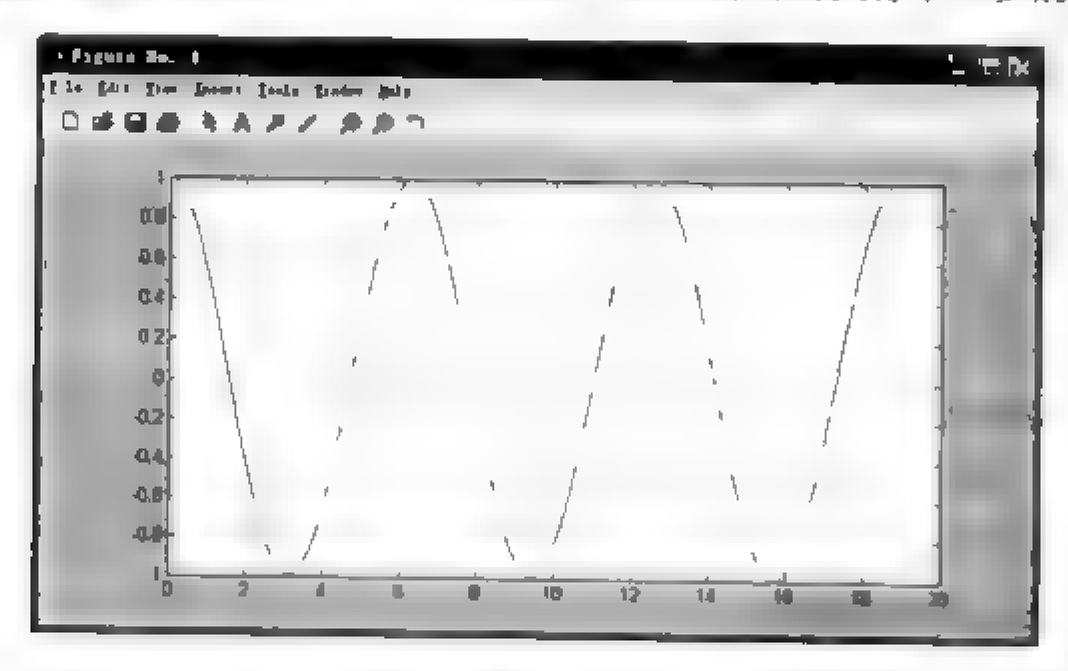


图 10-4　pause 命令的使用介绍 2

等待 10 秒后，系统将自动绘制出一条由上边的正弦曲线和余弦曲线相加的曲线，如图 10-5 所示。

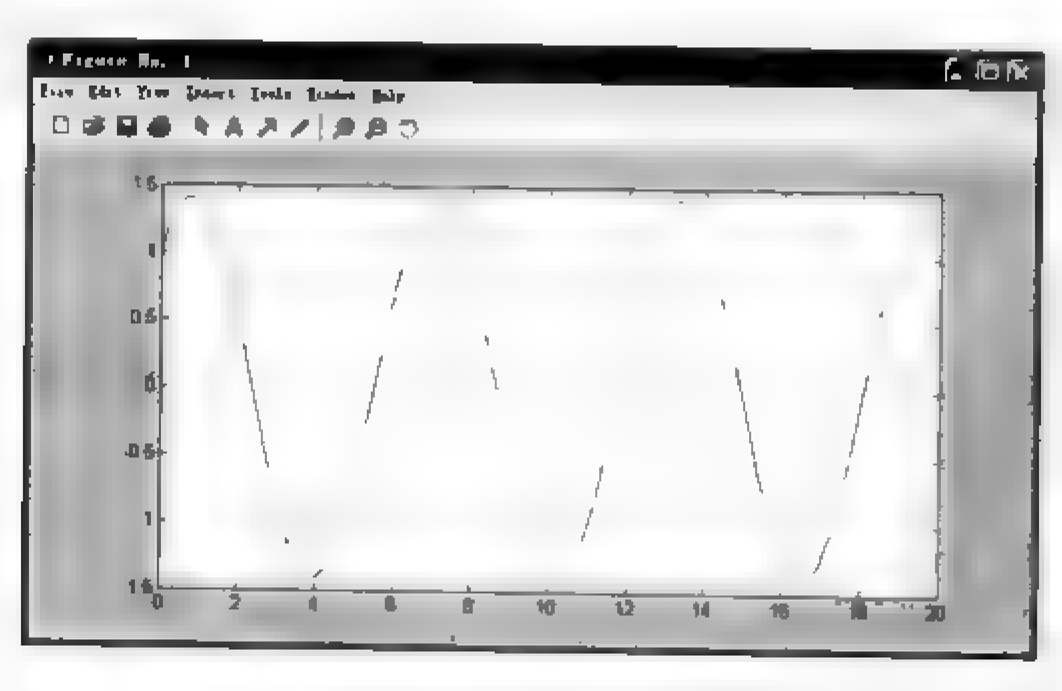


图 10-5　pause 命令的使用介绍 3

4. echo 命令

通常情况下，在执行 M 文件时，文件中的命令不会在命令窗口中显示。有时，为了调试或演示程序，需要这些命令在执行时可见，此时，可以使用 echo 命令来实现。其使用格式如下。

- echo on 命令打开脚本式文件的回应命令。
- echo off 命令将关闭回应命令。
- echo file on 命令在执行“file”文件时将其命令显示出来
- echo file off 命令关闭“file”文件的命令在执行时的显示。
- echo file 命令为文件在执行中的回应显示开关。
- echo on all 命令将显示该命令后所有执行文件的命令语句。
- echo off all 命令将关闭该命令后所有执行文件的命令语句。

5. 用户输入提示命令 input

input 命令用来提示用户从键盘输入数据、字符串或表达式，它有两者常用的使用形式，下面举例加以说明。

- *R* = *input*('*statement*') 命令运行后，将给出文字提示 *statement*，并等待键盘输入，用户可以输入任意 MATLAB 7.0 的表达式，返回的结果为 *R*，如果用户在没有输入任何东西的情况下返回，该命令将返回一个空矩阵。
- *R* = *input*('*statement*','*s*') 命令运行后，MATLAB 7.0 将等待输入，并把输入当成字符串给变量 R 赋值。输入的字符串中可能包括一条或是多条的“\n”语句，“\n”语句代表着跳到下一行的开头，用户想输入“\”时，可以输入“\\”。

例 10-20　input 命令的使用。

解：在命令窗口中输入如下语句，并按 Enter 键确认。

```
>> R = input('How many apples do you have?\n')
How many apples do you have?
```

此时，MATLAB 7.0 进入等待阶段，等待用户输入一个值，比如这里输入 45。

```
45
R =
    45
>>
```

输入 45 并按 Enter 键之后，MATLAB 7.0 将 45 赋值给 R。继续在 MATLAB 7.0 命令窗口中输入如下语句，并按 Enter 键确认。

```
R = input('Who do you love most?\n','s')
Who do you love most?
```

此时，MATLAB 7.0 进入等待阶段，等待用户输入一个字符串。继续在 MATLAB 7.0

命令窗口中输入如下语句，并按 Enter 键确认。

```
I don't know!
R =
I don't know!
>>
```

输入 I don't know!并按 Enter 键之后，MATLAB 7.0 将 I don't know!赋值给 R。

6. 请求键盘输入命令 keyboard

keyboard 命令与 input 命令的作用相似，当程序遇到次命令时，MATLAB 将暂时停止运行程序并处于等待键盘输入状态。处理完毕后，键入 R，程序将继续运行。在 M 文件中使用此命令，对程序的调试以及在程序运行中修改变量都很方便。

10.3　变量和函数种类

与其他语言一样，MATLAB 7.0 的变量有输入变量、输出变量和函数内使用的变量之分，而函数也有主函数、局部函数和子函数之分，本节将介绍这些不同种类变量和函数的特点。

10.3.1　函数变量及其作用域

在 MATLAB 7.0 语言中，变量可以分为输入变量、输出变量和函数内使用的变量。

输入变量相当于函数的入口数据，也是一个函数操作的主要对象，从某种意义上来说，函数的功能在于对输入变量进行一定的操作从而实现一定的功能。函数的输入变量为局部变量，函数对输入变量的一切操作和修改如果不依靠输出变量的话，将不会影响工作区间中该变量的值。

1. 变量的输入和输出规则

MATLAB 7.0 可以有任意数量的输入和输出变量。这些参数的特性和规则如下所示。

(1) 函数式 M 文件可以没有输入和输出变量。

(2) 函数可以用比 M 文件中的函数定义行所规定的输入输出变量更少的变量进行调用。但是不能用比规定的输入输出变量更多的变量进行调用。

(3) 在一次调用中所用到的输入和输出变量的个数可以通过分别调用函数 nargin 和 nargout 来确定。因为 nargin 和 nargout 是函数而不是变量，所以用户不能用诸如 nargin=nargin+pi 之类的语句对它们进行重新赋值。

(4) 当一个函数被调用时，输入变量并没有被复制到函数的工作区间中，但是它们的值在这个函数中是可读的。应当注意的是，如果输入变量的任何值被改变了，这个输入变

量组就被复制到了函数的工作区。这样，为了节省内存和提高速度，最好是将元素从大的数组中提取出来，然后再修改它们，而不是迫使整个数组都被复制到这个函数的工作区中。另外，对输入变量和输出变量使用相同的变量名将会使 MATLAB 7.0 立刻将输入变量的值复制到函数的工作区中。

(5) 如果一个函数声明了一个或者多个输出变量，但是用户在使用的时候又不想要输出参数，则只要不把输出变量赋值给任何变量就可以了；或者在函数结束之前用函数 clear 删除这些变量。

(6) 函数可以通过在函数声明中将 varargin 作为最后的输入参数，接受可变的任意个数的输入参数。varargin 是一个预先定义的单元数组，这个单元数组的第 i 个单元就是 varargin 出现的位置算起的第 i 个变量。

(7) 通过函数声明行中将 varargout 做为最后的输出变量，函数可以接受任意个数的变量形式的输出参数。varargout 也是一个预定义的单元数组，这个单元数组的第 i 个单元就是从 varargout 的出现位置算起的第 i 个变量。

(8) 函数 nargchk 和 nargoutch 分别提供了对有效的输入和输出变量个数的简单错误校验，因为如果函数调用的输入或者输出变量的个数多于函数定义中出现的个数，函数都自动地返回一个错误，因此虽然这些函数的作用有限，但是在一个函数定义声明了任意数目的输入变量和输出变量的时候却是非常有用的。

下面对其中的一些函数的使用方法举例予以说明。

例 10-21　nargin 函数的初级使用方法。

解：本程序实现如下功能，当调用过程时小于或等于一个变量时，系统提示错误的输入，当有两个变量时，程序将两个数相加，当有 3 个变量时，将前两个数相加并减去第 3 个。程序如下。

```
function d=nargintest(a,b,c)
if nargin<=1
      error('Not enough input arguments.')
elseif nargin==2
         d=a+b;
      elseif nargin==3
                  d=a+b-c;
end
```

此外，使用 nargin 可以查找函数输入变量的个数，例如，想要查找上述 nargintest 函数的参数个数，可以在 MATLAB 7.0 的命令窗口中输入如下命令。

```
>> nargin('nargintest')
ans =
      3
>>
```

例 10-22　nargin 函数的高级使用方法。

解：该程序用于在一个字符串中查找一小段指定的字符串，该段字符串为空格或是其他一些字符组成。如果输入一个变量的话，程序将默认所查找的字符串为空格，如果输入两个变量，程序将要求指定另一个定界符。同时，程序要求有两条输出语句。程序代码如下。

```
function [token, remainder] = strtok(string, delimiters)
% Function requires at least one input argument
if nargin < 1
    error('Not enough input arguments.');
end
token = []; remainder = [];
len = length(string);
if len == 0
    return
end

% If one input, use white space delimiter
if (nargin == 1)
    delimiters = [9:13 32]; % White space characters
end
i = 1;
% Determine where non-delimiter characters begin
while (any(string(i) == delimiters))
    i = i + 1;
    if (i > len), return, end
end
% Find where token ends
start = i;
while (~any(string(i) == delimiters))
    i = i + 1;
    if (i > len), break, end
end
finish = i - 1;
token = string(start:finish);
% For two output arguments, count characters after
% first delimiter (remainder)
if (nargout == 2)
    remainder = string(finish+1:end);
end
```

例 10-23　varargin 和 varargout 函数的使用举例。

本程序可以接收任意数量的两元素向量并在它们之间绘制直线。源程序如下。

```
function testvar(varargin)
for k = 1:length(varargin)
    x(k) = varargin{k}(1); % Cell array indexing
```

```
    y(k) = varargin{k}(2);
end
xmin = min(0,min(x));
ymin   min(0,min(y));
axis([xmin fix(max(x))+3 ymin fix(max(y))+3])
plot(x,y)
```

2. 全局变量

在 MATLAB 7.0 语言中，函数内部定义的变量都是局部变量，它们不被加载到工作区间中。有时，用户需要使用全局变量，这时要使用 global 函数来进行定义，而且在任何使用该全局变量的函数中都应加以定义，即使是在命令窗口也不例外。

例 10-24　全局变量的应用举例。

解：以 MALAB 的自带函数 tic 和 toc 为例加以说明。

```
function tic
%       TIC Start a stopwatch timer
%             TIC; any stuff; TOC
%       prints the time required.
%       See also: TOC, CLOCK.
global TICTOC
TICTOC = clock,

function t = toc
%       TOC Read the stopwatch timer.
%       TOC prints the elapsed time since TIC was used.
%       t = TOC; saves elapsed time in t, does not print.
%       See also: TIC, ETIME.
global TICTOC
if nargout < 1
     elapsed_time = etime(clock,TICTOC)
else
     t = etime(clock,TICTOC);
end
```

函数 tic 和 toc 构成了对 MATLAB 7.0 进行计时的一个简单秒表。当 tic 被调用的时候，它将 TICTOC 声明为全局变量，将当前的时间赋值给它，然后结束。此后，当 toc 函数被调用时，TICTOC 在 toc 工作区间中被声明为全局变量，这样就提供了对它的值的访问，然后就可以计算出所经过的时间。需要注意的是，变量 TICTOC 仅存在于函数 tic 和 toc 的工作区间中，它并不存在于 MATLAB 7.0 的工作区间中，除非在这个工作区间中也将 TICTOC 声明为全局变量。

3. 永久变量

除了通过全局变量共享数据外，函数式 M 文件还可以通过声明一个变量 persistent 来对函数中重复使用和递归调用的变量的访问进行限制，使用格式形如 persistent (X Y Z)。

永久变量与全局变量类似，但是它的范围被限制在声明这些变量的函数内部，不允许在其他的函数中对它们进行改变。只要 M 文件还在 MATLAB 7.0 的内存中，永久变量就存在。

10.3.2　函数的分类

1. 主函数

M 文件中的第一个函数就叫做主函数，前边章节中所引用的函数事实上都是主函数，主函数之后可以是任意数量的子函数，它们可以作为主程序的子程序。一般来说，在命令窗口或是其他的 M 文件只能调用主函数，调用的时候就是直接调用其函数名。

比如，函数 average 的 M 文件 average.m 如下。

```
function y = average(x)
% AVERAGE Mean of vector elements.
y = sum(x)/length(x);          % Actual computation
```

如果用户想求 12、60 和 42 这 3 个数的平均数，可以在命令窗口中调用 average 函数，格式为 average([12 60 42])。

通常都将程序名与 M 文件名保持一致，如果它们不一致的话，在调用的时候以 M 文件名为准。

2. 匿名函数

匿名函数提供了一种创建简单程序的方法，使用它用户可以不必每次都编写 M 文件。用户可以在 MATLAB 7.0 的命令窗口或是其他任意 M 文件和脚本文件中使用匿名函数。

匿名函数的格式如下所示。

```
fhandle = @(arglist) expr
```

其中，fhandle 是为该函数创建的函数句柄，有关函数句柄的内容将在下面的章节予以说明。@s 符号用于在 MATLAB 7.0 中创建函数句柄；arglist 是一些由逗号分隔的输入参数，这些参数将被传输到函数；而 expr 是函数的主体，它由一些 MATLAB 7.0 的执行语句组成。

例 10-25　匿名函数举例。

解：本例列举 3 个匿名函数的例子，分别有 0 个、1 个和两个输入参数。

首先介绍没有输入参数的匿名函数，只需用空格代替 arglist 即可。例如，编写求解当前时间的匿名函数如下所示。

```
>> t = @() datestr(now);
t()                              %函数的调用
ans =
14-Sep-2004 20:22:50
```

值得注意的是，调用该函数时，空格不能省略，否则， MATLAB 7.0 将对程序不予

计算，如下所示。

```
>> t
t =
      @() datestr(now)
>>
```

再介绍有一个输入参数的匿名函数，该函数用于求所输入参数的平方，编制并运行程序如下所示。

```
>> sqr = @(x) x.^2
sqr =
      @(x) x.^2
>> sqr(12)
ans =
    144
>>
```

最后介绍有两个输入参数的匿名函数，用户可以由此推导有多个参数的匿名函数，编制并运行程序如下所示。

```
>> sumAxBy = @(x, y) (14*x + 41*y)
sumAxBy
      @(x, y) (14*x + 41*y)
>> sumAxBy(3,7)
ans =
    329
>>
```

3. 嵌套式函数

在 MATLAB 7.0 中，可以在一个函数的内部定义一个或多个其他的函数，这些在内部定义的函数被称作嵌套式函数，应当注意的是，在嵌套式函数的内部也可以定义嵌套式函数。

定义嵌套式函数时，只需在另一个 M 文件的内部定义该函数即可，同其他 M 文件一样，嵌套式函数包含有 M 文件的所有基本部分。

值得用户注意的是，必须使用 end 命令来终止一个嵌套函数。嵌套式函数的使用格式如下。

```
function x = A(p1, p2)

    function y = B(p3)

    end
..
end
```

此外，可以在函数内定义多个平行的嵌套式函数，如下边的程序中，函数 A 下边设置

两个平行的嵌套函数 B 和 C。使用格式如下。

```
function x = A(p1, p2)
 .
    function y = B(p3)
    ...
    end

    function z = C(p4)
     ..
    end
 .
end
```

还可以在嵌套函数的内部再定义下一级的嵌套函数，如下边的程序中，函数 A 下边设置嵌套函数 B，B 下边又设置嵌套函数 C，其使用格式如下。

```
function x = A(p1, p2)
 ..
    function y = B(p3)
    . .
        function z = C(p4)
        ...
        end
     ..
    end
...
end
```

4. 子函数

与其他的高级语言一样，在 MATLAB 7.0 语言中也可以很方便地定义子函数，用来扩充函数的功能。在函数文件中题头定义的函数为主函数，而在函数体内定义的其他函数都被视为子函数。子函数只能为主函数或同一主函数下的其他子函数所使用。

例 10-26　编制一函数，要求任意输入两个数值后，用两个子函数分别求出它们的和与它们的绝对值的和，在将这两个和相乘。

解：编制程序如下：

```
function ch=zihanshu(x,y)                      %主函数
ch=zihanshu1(x,y)*zihanshu2(x,y);
function ch=zihanshu1(x,y)                     %子函数 1

ch=abs(x)+abs(y);
function ch=zihanshu2(x,y)                     %子函数 2
ch=x+y;
```

下面检验一下这个函数的正确性。

```
>> x=1;y=-9;
>> zihanshu(x,y)
ans =
    -80
>>
```

5. 局部函数

MATLAB 7.0 语言中把放置在目录 private 下的函数称为局部函数，这些函数只有 private 目录的父目录中的函数才可以调用，其他目录下的函数不能调用。

局部函数与子函数所不同的是，局部函数可以被其父目录下的所有函数所调用，而子函数则只能被其所在的 M 文件的主函数所调用。所以，局部函数在可用的范围上大于子函数；在函数编辑的结构上，局部函数与一般的函数文件的编辑相同，而子函数只能在主函数文件中编辑。

当在 MATLAB 7.0 的 M 文件中调用函数时，将首先检测该函数是否为此函数的子函数；如果不是的话，再检测是否为可用的局部函数；如果仍然不是的话，再检测该函数是否为 MATLAB 7.0 搜索路径上的其他 M 文件。

10.3.3　函数句柄

函数句柄提供了一种间接访问函数的手段，用户可以很方便地调用其他函数；提供函数调用过程中的可靠性；减少程序设计中的冗余；同时可以在使用函数的过程中保存函数相关的信息，尤其是关于函数执行的信息。

1. 函数句柄的创建

函数句柄的创建比较简单，使用格式形如：$fhandle = @functionname$，其中 $fhandle$ 为所创建的函数句柄，$functionname$ 为所创建的函数。

给匿名函数创建函数句柄如前边所述，$sqr = @(x)x.\hat{}2$ 即给函数 $x.\hat{}2$ 创建了函数句柄。

当需要创建函数句柄列时，可以使用单元数组，如下例所示。

```
>> trigFun = {@sin, @cos, @tan};
>> plot(trigFun{2}(-pi:0.01:pi))
>>
```

运行该程序生成如图 10-6 所示。

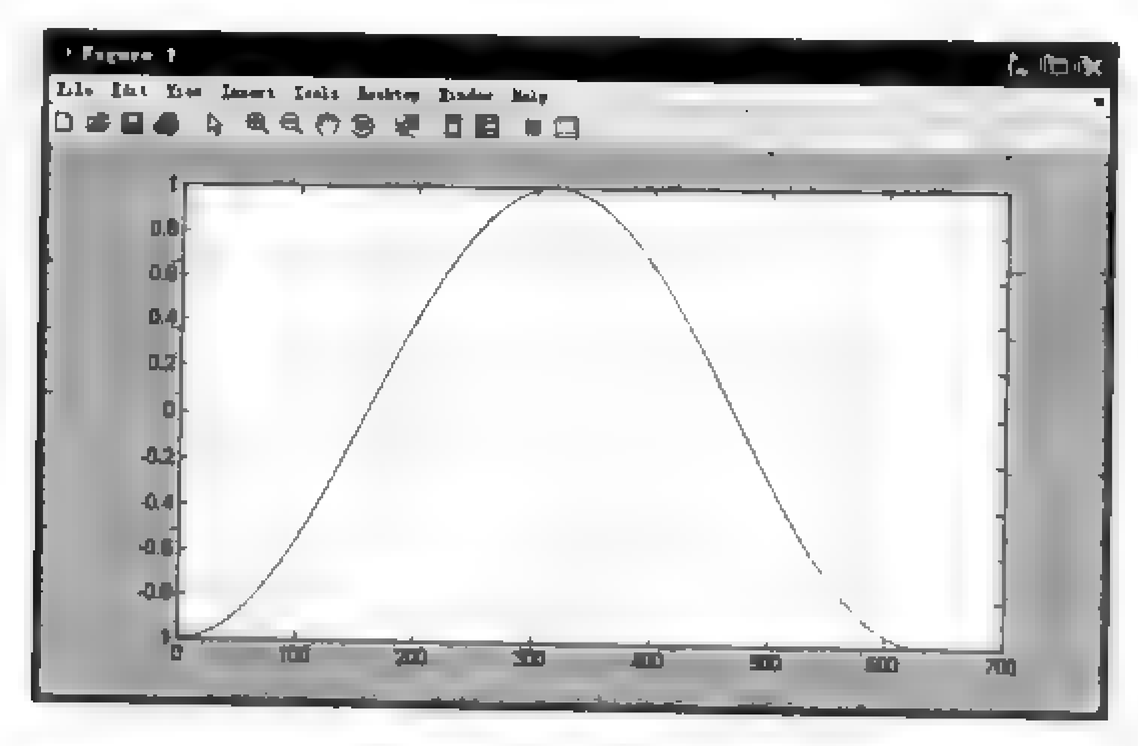


图 10-6　函数句柄列的应用

在给出不在当前路径下的函数创建函数句柄时，将产生无效的函数句柄。

2. 函数句柄的调用

函数句柄的调用比较简单，用户可以通过下例来掌握函数句柄的调用方法。

例 10-27　本例调用一个函数 plotfhandle，传递一个 MATLAB 7.0 内置的 sin 函数给该函数，然后，由 plotfhandle 调用 plot 函数，将数据和 sin 的函数句柄传递给它。plot 函数调用与该句柄相关的函数来绘制曲线。

```
function x = plotFHandle(fhandle, data)
plot(data, fhandle(data))
```

在命令窗口中调用程序如下。

```
plotFHandle(@sin, -pi:0.01:pi)
```

所得图形如图 10-7 所示。

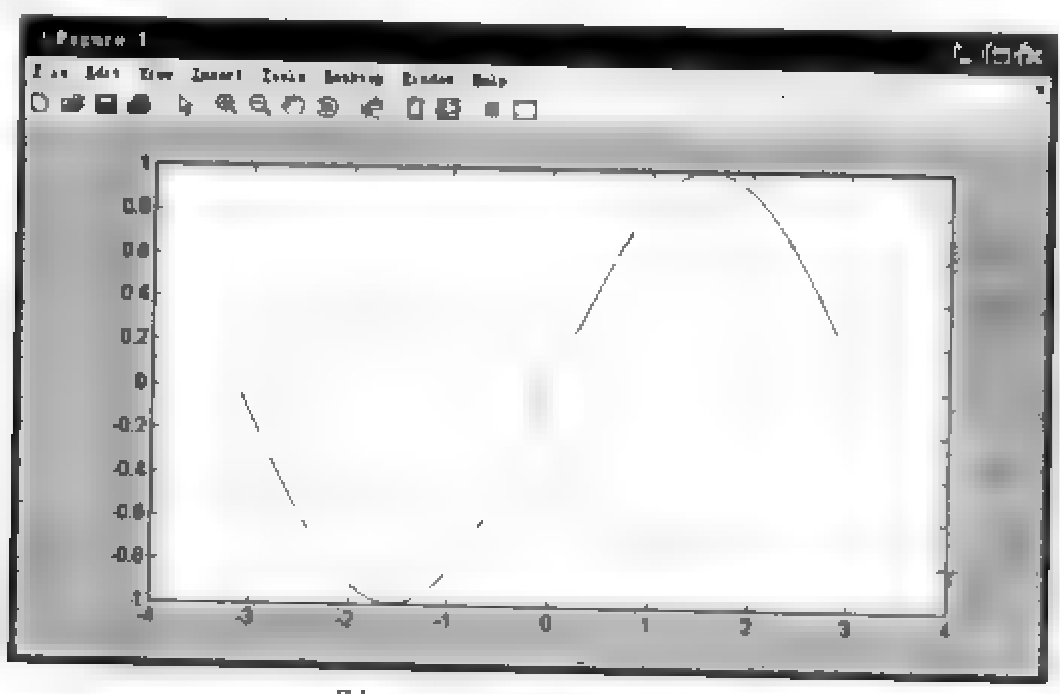


图 10-7　函数句柄的调用

3. 函数句柄的操作

MATLAB 7.0 提供了要一些对函数句柄进行操作的函数，如表 10-2 所示。

表 10-2　函数句柄的操作

函　数　名	功 能 描 述
functions	返回函数句柄的相关信息
func2str	根据函数句柄创建一个函数名的字符串
str2func	由一个函数名的字符串创建一个函数句柄
save	从当前工作区间向 M 文件保存函数句柄
load	从一个 M 文件中向当前工作区间调用函数句柄
isa	判断一个变量是否包含由一个函数句柄
isequal	判断 2 个函数句柄是否为某一相同函数的句柄

下面对其中的几个函数予以简单介绍。

例 10-28　functions 函数使用举例。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> functions(sumAxBy)
ans =
        function: '@(x, y) (14*x + 41*y)'
            type: 'anonymous'
            file: ''
       workspace: [1x1 struct]
>>
>> functions(sqr)
ans =
        function: '@(x) x.^2'
            type: 'anonymous'
            file: ''
       workspace: [1x1 struct]
>>
```

例 10-29　func2str 和 str2func 函数的使用。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> funhandleone=str2func('magic')
funhandleone =
    @magic
>> func2str(funhandleone)
ans =
magic
>>
```

例 10-30　isa 和 isequal 函数的使用。

解：isa 函数的使用格式为 *isa(OBJ,' function handle')*；isequal 函数的使用格式为 *isequal(A,B…)*。

首先是关于 isa 函数的示例。

```
>> isa(funhandleone,'function_handle')
ans =
     1
>> isa(funhandle,'function_handle')
??? Undefined function or variable 'funhandle'.
>>
```

下面是关于 isequal 函数的示例。

```
>> isequal(funhandleone,@magic)
ans =
     1
>>
```

10.4　程序设计的辅助函数

在编写 MATLAB 7.0 程序的过程中，有一些辅助函数对用户具有特殊的意义，其中执行函数、容错函数和时间控制函数等对程序的调试和优化具有特别重要的意义。本节将具体介绍这方面的知识。

10.4.1　执行函数

MATLAB 7.0 提供的执行函数主要有 eval、feval 和 run 等，如表 10-3 所示。

表 10-3　执行函数及其功能

函　数　名	功 能 描 述
assignin	在 MATLAB 7.0 工作区间中分配变量
builtin	外部加载调用内置函数
eval	字符串调用函数
evalc	执行 MATLAB 7.0 的表达式
evalin	计算工作区间中的表达式
feval	字符串调用 M 文件
run	运行脚本文件

下面对其中的几个函数予以简单介绍。用户可以由此而推知其他函数的使用方法。

1. eval 函数的使用方法

- $eval(e)$ 命令计算表达式 e 的值，e 为任意有效的 MATLAB 7.0 语言。
- $[a1,a2,a3,...]=eval(function(b1,b2,b3,...))$ 命令计算具有参数 $b1$、$b2$、$b3$……的函数 *function* 并返回 $a1$、$a2$ 和 $a3$ 等参数。

例 10-31　eval 函数的使用。

解：以下程序段产生 4 个魔术矩阵并将其命名为 M1 到 M4。

```
for n = 1:4
    magic_str = ['M',int2str(n),' = magic(n)'];
    eval(magic_str)
end
```

程序的运行结果如下。

```
M1 =
     1
M2 =
     1     3
     4     2
M3 =
     8     1     6
     3     5     7
     4     9     2
M4 =
    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
>>
```

以下程序段将对三维矩阵执行 size 命令，并以变量 d1、d2 和 d3 返回矩阵维数。

```
A = magic(4);
A(:,:,2) = A';
[d1,d2,d3] = eval('size(A)')
```

程序的运行结果如下。

```
d1 =
     4
d2 =
     4
d3 =
```

```
        2
>>
```

2. feval 函数的使用方法

在某些情况下，用户需要把一个函数的字符串名传递给一个函数进行计算。例如，MATLAB 7.0 语言的很多数值分析函数都需要利用函数的名字进行计算。MATLAB 7.0 提供了函数 feval 来完成这种计算。

- $feval(F,x1,...,xn)$ 命令计算 F 关于参数 x1、…… xn 的值，F 为函数名或是函数句柄。

  例如，如果 $F=@foo$，那么 $feval(F,9,54)$ 等价于 $foo(9,54)$。
- [y1,...,yn] = *FEVAL(F,x1,...,xn)*命令返回多个参数。

例 10-32　feval 函数的使用方法。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> fhandle = @deblank;
ff = functions(fhandle);
>> feval(fhandle, {'string     ','with   ','blanks            '})
ans =
    'string'      'with'      'blanks'
>>
```

10.4.2　容错函数

程序设计的好坏在很大程度上取决于其容错能力的大小。MATLAB 7.0 语言提供了相应的报错及警告函数 error、warning、lasterr、lastwarn 以及 errortrap on/off 等，可以很方便地实现这方面的功能。下面对其中的 error 和 warning 函数分别予以介绍。

1. 报错函数 error 的使用

函数 error 可以在命令窗口中显示错误信息，以提示用户程序中发生了错误，其调用格式如下。

- 当调用 M 文件的时候触发了函数 error，$error('m')$ 命令将中断程序的运行，将控制权返回键盘，并返回包含 m 的错误信息，当 m 为空字符串时本命令将不做任何操作。
- $error('m',a1,a2,...)$ 命令使用诸如 sprintf 类的函数返回带格式的错误信息。
- $error('m_id','m')$ 命令给错误信息附加上一个特定的信息标识，该标识有利于用户更好地找出错误的来源。
- $error('m_id','m',a1,a2,...)$ 返回带格式的错误信息，并给错误信息附加了特定的信息标识。

例 10-33　函数 error 的使用。

解：首先编制 M 文件如下，当输入参数的数目不等于 2 时，函数将给出出错信息。

```
function foo(x,y)
if nargin ~= 2
        error('Wrong number of input arguments')
end
```

运行程序如下。

```
>> foo(1)
??? Error using ==> foo                    %由于输入参数为 1，所以触发 error 函数。
Wrong number of input arguments
>>
```

2. 警告函数 warning 的使用

函数 warning 的使用方法与函数 error 类似，不同的是，函数 error 生成错误信息后，程序将停止运行并等待用户做进一步的操作，而函数 warning 在触发后只提出警告信息但不停止程序的运行。其调用格式如下。

- *warning('m')* 返回包含 *m* 的警告信息。
- *warning('m',a1,a2,...)* 命令使用诸如 sprintf 类的函数返回带格式的警告信息。
- *warning('m_id','m')* 命令给警告信息附加上一个特定的信息标识，该标识有利于用户更好地找出警告的来源。
- *warning('m_id','m',a1,a2,...,an)* 返回带格式的错误信息，并给错误信息附加了特定的信息标识。
- *s = warning('m_id','m')* 命令是一条警告控制语句，它可以使用户指示 MATLAB 7.0 将对该警告采取什么样的操作。

例 10-34　警告函数 warning 的使用。

解：首先编制 M 文件如下，当输入参数的数目等于 2 时，程序计算两个数的乘积；当输入参数的数目等于 1 时，函数将给出警告信息并计算它的平方；当输入参数不是 1 和 2 时，给出错误信息。程序如下。

```
function z=warning1(x,y)
if nargin==2
      z=x*y,
elseif nargin==1
        {
      warning('you should input 2 arguments')
        }
      else
      error('Wrong number of input arguments')
end
```

程序的运行结果如下。

```
>> warning1(1)
Warning. you should input 2 arguments
> In warning1 at 6
ans
      {''}
>> warning1(1,2)
ans =
        2
>> warning1(1,2,3)
??? Error using ==> warning1
Too many input arguments.
>>
```

10.4.3 时间运算函数

MATLAB 7.0 提供了许多函数来处理时间，用户可以对日期和时间进行数学运算，打印日历以及找到指定的某天。同时，MATLAB 7.0 也提供了计时函数来计算程序运行所消耗的时间，这样，用户可以使用它们做为程序优化的依据。MATLAB 7.0 中常用的时间控制函数如表 10-4 所示。

表 10-4 常用的时间控制函数及其功能

函数名	功能描述
clock	以向量的形式显示当前的时间和日期
date	以字符型显示当前的日期
now	以数值型显示当前的时间和日期
calendar	显示当月的日历表
weekday	显示当前日期对应的星期表达
eomday	给出指定年月的当月最后一天
datetick	指定坐标轴的日期表达形式
datevec	转换为向量形式显示日期
datenum	转换为数值型格式显示日期
datestr	转换为字符型格式显示日期
tic	计时的开始函数
toc	计时的结束函数
cputime	以 CPU 的运行时间方式进行计时
etime	计算 2 个时刻的时间差

下面，对上边的时间函数分类予以介绍。

1. 当前日期和时间

这类函数主要包括 clock、date、now、calendar 和 weekday 等。下面通过实例对它们予以介绍。

例 10-35　MATLAB 中显示当前日期和时间。

解：MATLAB 7.0 中输入上述时间函数，并按 Enter 键确认。

```
>> T=clock
T =
   1.0e+003 *
     2.0040     0.0090     0.0160     0.0140     0.0580     0.0092
>>
```

以上是以向量形式表示的本书这一部分编写时所处的时间，即 2004 年 9 月 16 号 14 时 58 分第 9.2 秒。

```
>>date
ans =
16-Sep-2004
>>
```

以上是编写本身这一部分的日期。

```
>> now
ans =
   7.3221e+005
>>
```

以上是以双精度数据表示的当前的日期和时间。

```
>>calendar
                  Sep 2004
     S     M    Tu     W    Th     F     S
     0     0     0     1     2     3     4
     5     6     7     8     9    10    11
    12    13    14    15    16    17    18
    19    20    21    22    23    24    25
    26    27    28    29    30     0     0
     0     0     0     0     0     0     0
>>
```

以上是当前月的日历。

```
>> [n,s] = weekday(728647)
n =
```

```
          2.00
s
Mon
>> [n,s] = weekday('19-sep-2004')
n =
          1.00
s
Sun
>>
```

以上是函数 weekday 的使用方法。

```
>> y = 1900:1999;
>>E = eomday(y,2*ones(length(y),1)');
>> y(find(E==29))
ans =
  Columns 1 through 6
      1904.00      1908.00      1912.00      1916.00      1920.00      1924.00
  Columns 7 through 12
      1928.00      1932.00      1936.00      1940.00      1944.00      1948.00
  Columns 13 through 18
      1952.00      1956.00      1960.00      1964.00      1968.00      1972.00
  Columns 19 through 24
      1976.00      1980.00      1984.00      1988.00      1992.00      1996.00
>>
```

以上是利用函数 eomday 来查找从 1900 到 1999 年有哪些年份的 2 月有 29 天。

2. 日期转换函数

通常，对于带时间的数学都涉及到将时间转换为日期数字格式，对时间执行标准的数学运算，然后将结果数字再转换为日期格式。MATLAB 7.0 提供了 3 种日期格式，即双精度日期数字、不同形式的日期(字符)字符串以及数值型的日期向量。实现它们之间的相互转换主要由函数 datestr 、datevec 和 datenum 等来实现。它们的使用格式如下。

(1) datestr 函数

- $str = datestr(DT)$ 将由 datenum 定义的数值型日期转换为字符型日期。
- $str = datestr(DT, dateform)$ 将由 datenum 定义的数值型日期转换为带特定格式的字符型日期。dateform 必须是数字 0、1、2、6、13、14、15、16 或 23 中的一个。
- $str = datestr(DT, dateform, P)$ 使用特定的年做为起始年，默认的起始年是当前年减去 50。
- $str = datestr(..., 'local')$ 命令返回一个本地形式的字符型日期，默认的形式为美国英语。

datestr 函数中使用的日期格式如表 10-5 所示。

表 10-5　datestr 中使用的日期格式

编　号	格　式	举　例
0	'dd-mmm-yyyy HH:MM:SS'	01-Mar-2000 15:45:17
1	'dd-mmm-yyyy'	01-Mar-2000
2	'mm/dd/yy'	03/01/00
3	'mmm'	Mar
4	'm'	M
5	'mm'	03
6	'mm/dd'	03/01
7	'dd'	01
8	'ddd'	Wed
9	'd'	W
10	'yyyy'	2000
11	'yy'	00
12	'mmmyy'	Mar00
13	'HH:MM:SS'	15:45:17
14	'HH:MM:SS PM'	3:45:17 PM
15	'HH:MM'	15:45
16	'HH:MM PM'	3:45 PM
17	'QQ-YY'	Q1-96
18	'QQ'	Q1
19	'dd/mm'	01/03
20	'dd/mm/yy'	01/03/00
21	'mmm.dd,yyyy HH:MM:SS'	Mar.01,2000 15:45:17
22	'mmm.dd,yyyy'	Mar.01,2000
23	'mm/dd/yyyy'	03/01/2000
24	'dd/mm/yyyy'	01/03/2000
25	'yy/mm/dd'	00/03/01
26	'yyyy/mm/dd'	2000/03/01
27	'QQ-YYYY'	Q1-1996
28	'mmmyyyy'	Mar2000
29(ISO 8601)	'yyyy-mm-dd'	2000-03-01
30(ISO 8601)	'yyyymmddTHHMMSS'	20000301T154517
31	'yyyy-mm-dd HH:MM:SS'	2000-03-01 15:45:17

(2) datenum 函数

datenum 函数将字符型日期和向量型日期转换为数值型日期。

当 DT 是字符型日期，并且符合如上所述的第 0、1、2、6、13、14、15、16 或 23 种格式时，datenum 可以实现如下操作。

- $N = datenum(DT)$ 命令将字符型或是向量型日期 DT 转换为数值型日期，其中的年数必须在以当前年为中心的 100 年内。
- $N = datenum(DT,P)$ 命令指定一个起始年，默认形式为当前年减去 50 年。
- $N = datenum(DT,F)$ 命令在进行转换操作时使用特定的日期格式 F。
- $N = datenum(DT,F,P)$ 将上述两个命令的功能结合起来。

(3) datevec 函数

datevec 函数用于将其他日期形式转换为向量形式显示日期，其使用格式如下。

- $V = datevec(DT)$ 命令将数值型日期和字符型日期转换为向量型日期。该向量包含如下元素——year、month、day、hour、minute 和 second。其中前 5 个元素为整数。也可以将包含 N 个数值型日期的向量转换为一个 N×6 阶矩阵。
- $V = datevec(DT,P)$ 命令指定一个起始年，默认形式为当前年减去 50 年。下面通过实例对它们予以介绍。
- $V = datevec(DT,F)$ 命令在进行转换操作时使用特定的日期格式 F。
- $V = datevec(DT,F,P)$ 将上述两个命令的功能结合起来。
- $[Y,M,D,MI,S] = datevec(DT)$ 命令将日期元素以单独的变量返回。当创建用户自己的日期向量时，可以使用户不必要将日期作为一个整体进行操作。

例 10-36　日期转换函数的使用举例。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> t=datestr(now)
t =
16-Sep-2004 16:30:26
>> datestr(now, 2)
ans =
09/16/04
>> datestr(now, 'dd.mm.yyyy')
ans =
16 09 2004
>> datestr(datenum('24.01.2003', 'dd.mm.yyyy'), 2)
ans =
01/24/03
>> datestr(now,23)
ans
09/16/2004
>> datestr(now,16)
ans =
```

```
 4:31 PM
>>
```

以上部分介绍了 datestr 函数的使用方法，下边的程序将介绍 datenum 函数的使用方法。继续在 MATLAB 7.0 的命令窗口中输入如下命令，并按 Enter 键确认。

```
>> n = datenum('09/16/04')
n =
        732206
>> n = datenum('01/24/03')
n =
        731605
>> n = datenum('19-May-2001')
n =
        730990
>> n = datenum(2001, 12, 19)
n −
        731204
>> format bank
>> n = datenum([2001 5 19 18 0 0])
n =
      730990.75
>> n = datenum('12-june-12')
n ÷
      735032.00
>> n = datenum('12-june-12', 1900)
n =
      698507.00
>> n = datenum('19.05.2000', 'dd mm.yyyy')
n =
      730624.96
>>
```

以上部分介绍了 datenum 函数的使用方法，下边的程序将介绍 datevec 函数的使用方法。继续在 MATLAB 7.0 的命令窗口中输入如下命令，并按 Enter 键确认。

```
>> datevec('12/24/1984 12 45')
ans
        1984.00        12.00        24.00        12.00        45.00        0
>> t = datenum('12/24/1984 12:45')
t =
      725000 53
>> [y, m, d, h, mi, s]   datevec('12/24/1984 12:45');
>> sprintf('Date  %d/%d/%d     Time: %d:%d\n', m, d, y, h, mi)
ans
```

```
Date: 12/24/1984     Time: 12:45
>> datevec('19.05.2003', 'dd.mm.yyyy')
ans =
        2003.00          5.00          19.00          0          0          0
>>
```

3. 计时函数

MATLAB 7.0 的计时函数主要有 tic、toc、cputime 和 etime 等，计时函数可以定量地计算完成指定程序所消耗的时间资源，因此可以作为比较程序优劣的一个重要标准。

下面对这些函数的功能做简单介绍。

(1) tic 和 toc 函数

这两个函数一般配合使用，tic 表示计时的开始，toc 表示计时的结束。使用格式如下。

```
tic
    任意表达式
toc
t = toc
```

下面举例予以说明。

例 10-37　tic 和 toc 函数的使用举例。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> for n = 1:100
    A = rand(n,n);
    b = rand(n,1);
    tic
    x = A\b;
    t(n) = toc,
end
>> plot(t)
>> t=toc
t =
          3.44                    %程序的运行耗时 3.44 秒
>>
```

生成的图形如图 10-8 所示。

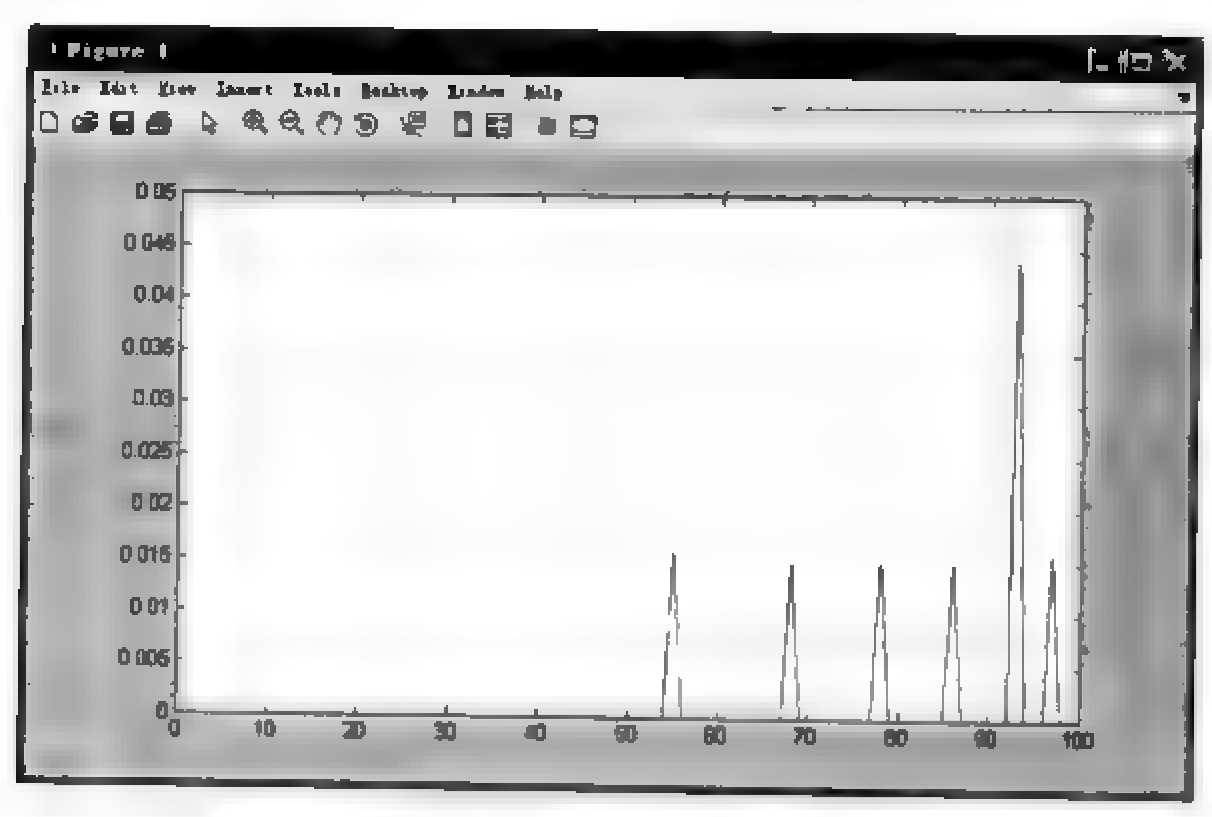


图 10-8　tic 和 toc 函数的使用

(2) cputime 函数

cputime 函数返回从调用该函数起所用的总的 CPU 时间，单位以秒计算。使用格式如下。

```
t = cputime;
```

任意程序或表达式

```
e = cputime-t
```

下面举例予以说明。

例 10-38　cputime 函数的使用举例。

```
>> t = cputime,
>> A=magic(4)*rand(4)
A =
        13.92        11.18        5 87        15 41
        13.53        16.57        13.64       14.08
        13.61        14.07        10.59       15 08
        13.68        18.67        15.00       12.40
>> e=cputime-t
e =
        0 61
>>
```

(3) etime 函数

e = etime(t2,t1)命令返回向量 t1 和 t2 之间的时间段，t1 和 t2 必须含有由 clock 函数返回的 6 个元素。即 T = [Year Month Day Hour Minute Second]。

下边举例予以说明。

例 10-39　etime 函数的使用举例。

```
>> x = rand(2048,1);
>> t = clock;
>> fft(x);
>> etime(clock,t)
ans =
         14.53
>>
```

10.5　程序的调试和优化

用户在编制程序的过程中，不可避免会遇到错误，尤其是对初学者来说更是如此，此外，MATLAB 7.0 还提供了一个帮助用户提高 M 文件的执行速度的工具。在分析一个 M 文件的执行方面，MATLAB 7.0 提供能够标识出代码的哪一行花费的运行时间最长。

在调试好程序以后，进一步需要做的工作是对它进行优化，经过优化的程序与没经过优化的程序相比，速度会有很大的区别，所以对编制大型的应用程序来说，程序的优化具有很重要的意义。本节将介绍几种 MATLAB 7.0 提供的调试程序的方法和函数，并讲述优化程序的一些方法。

10.5.1　程序的调试

在 MATLAB 7.0 的程序编辑器中提供了相应的程序调试功能，本节将简要介绍调试程序的具体方法。

1. 程序的错误种类

在 MATLAB 7.0 的表达式中可能存在两种类型的错误，即语法错误和运行错误，下面分别予以介绍。

(1) 语法错误

语法错误发生在 M 文件程序代码的生成过程中，一般是由函数参数输入类型有误或是矩阵运算阶数不符等引起。对于语法错误，MATLAB 7.0 会即到标记出这些错误，并返回所遇到的错误的类型以及该错误在其所在 M 文件中的行数。用户利用这些信息可以很方便地查找相关错误的位置和类型。但在 GUI 回调字符串中的语法错误却是一个例外，这些错误在 GUI 运行的过程中，知道字符串本身被执行了才会检测到错误。

例 10-40　语法错误举例。

解：本例将介绍几种常用的语法错误，在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> min(x,y)
```

```
??? Undefined function or variable 'x'.                    %使用没有声明的变量
>>
```

继续在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> x=0,y=-9;
>> y=minimun(x,y)
??? Undefined command/function 'minimun'.                  %使用没有声明的函数
>>
```

继续在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> A=ones(3)
A =
     1     1     1
     1     1     1
     1     1     1
>> B=magic(4)
B =
    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
>> C=A*b
??? Undefined function or variable 'b'.                    %使用没有声明的变量
>> C=A*B
??? Error using ==> mtimes
Inner matrix dimensions must agree.                        %矩阵的阶数不相符合
>>
```

(2) 运行错误

运行错误一般指在程序运行过程中，出现溢出或是死循环等异常现象。MATLAB 7.0 有时能将它们标识出来，但这些错误通常情况下都很难发现。出现运行错误后，MATLAB 7.0 把控制权返回给命令窗口和 MATLAB 7.0 工作区间。此时用户无法访问错误发生的函数工作区，因此无法通过查询这个函数的工作区来试图隔离出现的问题。

一般来说，最常见的运行错误发生在某项运算的结果或是中间结果导致了空数组、NaN 值或是 Inf 值。对空数组的寻址总会导致错误，因为空数组的维数为 0。函数 find 就是一个常常得到空数组结果的函数，如果 find 函数的空数组输出被用来索引某个其他数组，返回的结果数组也将是空数组。也就是说，空矩阵将导致空矩阵。而对于 NaN 值来说，因为所有对 NaN 的运算都会返回 NaN，此时，如果在运算过程中出现了 NaN 值，会出现许多用户意想不到的结果。因此，在可能出现空数组、NaN 值或是 Inf 值的地方，用户最好使用逻辑函数 isempty、isnan 和 isinf 来实现某项默认的操作。

例 10-41　运行错误的处理。

解：编制 M 文件如下。

```
function d=yunxingcuowu(a,b,c)
d=a/b;
if isinf(d)                                %对可能产生的 inf 值做出判断
      sprintf('b should not be 0')
return
else
   d=d*c;
end
```

为了检验该程序，可以运行程序如下，在 MATLAB 7.0 命令窗口中输入如下数据，并按 Enter 键确认。

```
>> d=yunxingcuowu(1,0,0)
Warning: Divide by zero.
> In yunxingcuowu at 2
ans =
b should not be 0
d =
    Inf
>>
```

2. 错误的识别和程序调试

(1) 识别错误的基本技巧

一般来说，语法错误比较容易识别，因为出现语法错误时，MATLAB 7.0 会提示相应的错误信息，以方便用户的检查和定位。但是，运行错误往往比较难以识别，因为发生运行错误时，系统会自动终止对 M 文件的调用，这样就关闭了函数的工作区间，无法获取需要的数据信息。MATLAB 7.0 提供了好几种识别程序错误的方法，对于简单的问题，使用下面的一种或几种方法可以方便地求解。

- 将函数中输出关键值的行的分号(;)去掉，这样，这些运算的中间结果将在命令窗口中予以显示，用户可以据此来检查中间结果的正确性。
- 在函数中添加一些语句，用来显示用户认为很重要的变量的值。
- 使用 keyboard 命令中断程序，该命令实现函数工作区间和命令窗口工作区间的交互，从而获得用户所需要的信息，使用该命令后，程序将处于调试状态，此时命令窗口的提示符由“>>”变为“K>>”，用户可以进行相应的操作。
- 在函数头前加“%”，这样就将函数式 M 文件变为脚本式 M 文件，而脚本式 M 文件运行时，其工作区间就是 MATLAB 7.0 的工作区间，这样在出现错误的时候就可以查询这个工作区间。

第12章　图 形 处 理

与数值计算和符号计算相比，图形的可视化技术是数学计算人员所追求的更高级的一种技术，因为对于数值计算和符号计算来说，不管计算的结果是多么的准确，人们往往无法直接从大量的数据和符号中体会它们的具体含义。而图形处理技术则给人们提供了一种更直接的表达方式，可以使人们更直接、更清楚地了解事物的结果和本质。MATLAB 7.0 语言除了有强大的矩阵处理功能之外，它的绘图功能也是相当强大的。本章将主要介绍 MATLAB 7.0 的图形处理功能，包括基本的绘图命令、图形的简单控制、图形窗口的编辑以及图形的高级控制等。

12.1　基本的绘图命令

MATLAB 7.0 语言在绘图方面的功能非常全面，可以很方便地绘制二维、三维甚至多维图形。本节将主要介绍使用 MATLAB 7.0 语言进行基本的图形绘制，并介绍一些简单的图形操作命令。

12.1.1　图形窗口简介

MATLAB 7.0 语言的绘图函数和工具将所绘制的图形在图形窗口中显示出来，该窗口将与 MATLAB 7.0 的主窗口隔离出来。如图 12-1 标识了图形窗口的主要部分。

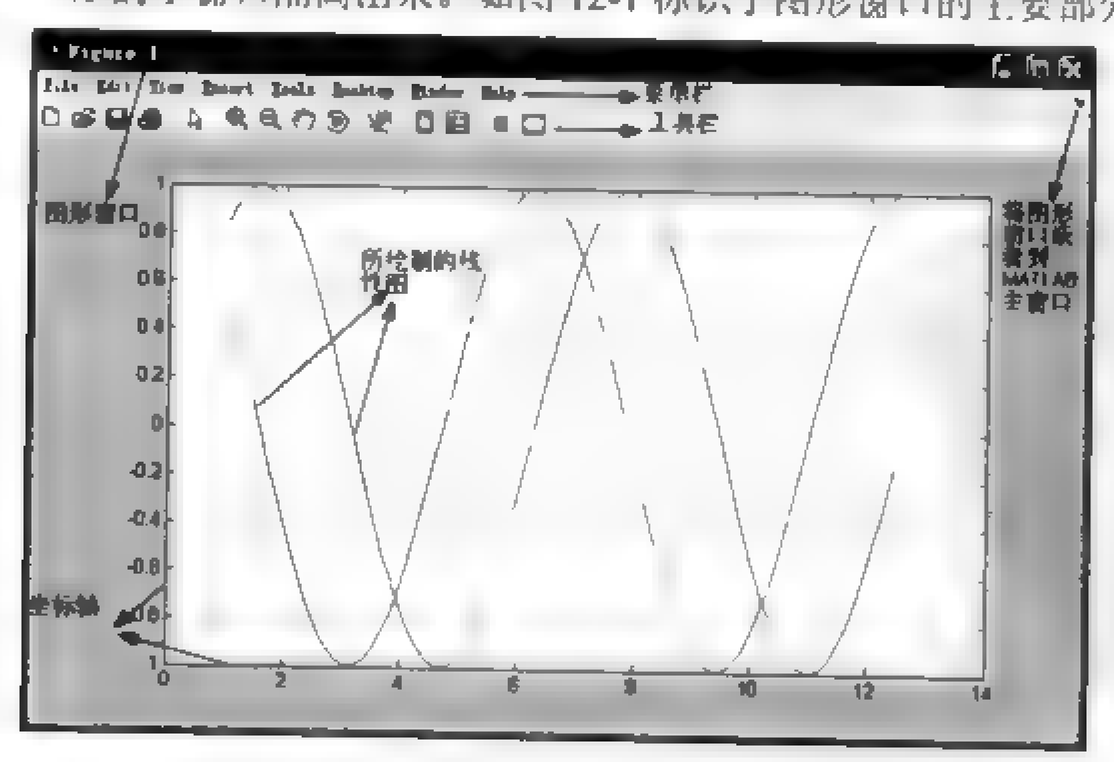


图 12-1　图形窗口

例 10-42 在 M 文件中使用断点进行操作。

解：编制 M 文件如下。

```
function z = buggy(x)
n = length(x);
z = (1:n)./x,
```

继续在 MATLAB 7.0 中输入如下命令，并按 Enter 键确认。

```
>> dbstop in buggy                    %在可执行程序段的第一行设置断点
>> buggy(2:5)
K>>
```

此时，命令窗口就显示出键盘提示符号“K>>”，并返回编辑窗口，在编辑窗口中设置断点的地方 MATLAB 7.0 用红点和绿色的箭头予以标出。用户可以查询函数工作区间，并做出相应的修改，如图 10-9 所示。

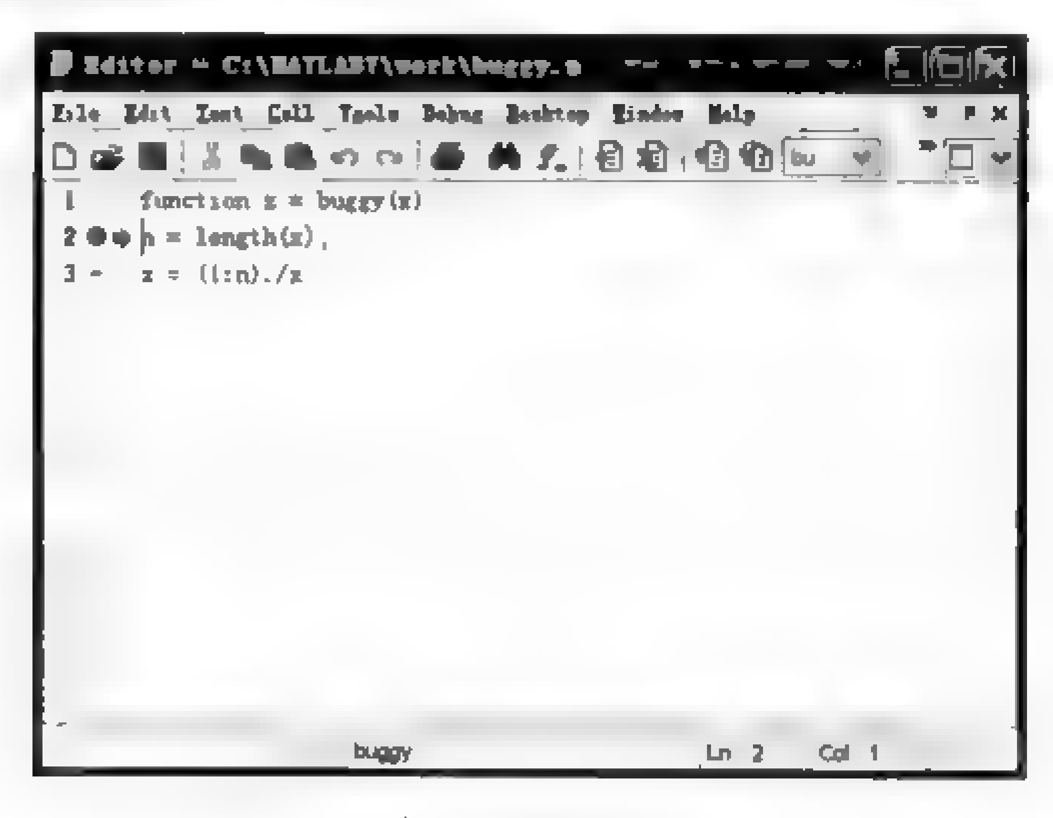


图 10-9 断点设置 1

MATLAB 7.0 中输入 return 命令后，程序将恢复运行。

```
K>> return                                        %return 命令恢复函数运行
ans =
    0.5000    0.6667    0.7500    0.8000          %运行结果
>>
```

继续在 MATLAB 7.0 中输入如下命令，并按 Enter 键确认。

```
>>dbstop if error                                 %在函数错误处设置断点
>> buggy(magic(3))
K>> return
```

```
??? Error using ==> rdivide                          %返回的错误信息
Matrix dimensions must agree.

Error in ==> buggy at 3
z  (1 n)  x,
K>>
```

继续在 MATLAB 7.0 中输入如下命令，并按 Enter 键确认。

```
>>dbstop if naninf                                   %当出现 NaN 或 inf 值时设置断点
>> buggy(0:2)
K>> return                                           %返回的错误信息和如图 10-10 所示
Warning Divide by zero.
> In buggy at 3
NaN/Inf breakpoint hit for buggy on line 3.
3     z  (1 n)./x,
java.util.NoSuchElementException
  at java.util.StringTokenizer.nextToken(Unknown Source)
  at com.mathworks.mlwidgets.workspace.MATLAB
7 0WorkspaceModel$QVCO.completed(MATLAB 7 0WorkspaceModel java 261)
  at com.mathworks jmi MATLAB 7 0$CompletionHandler messageReceived(MATLAB
7 0.java 2079)
  at com mathworks.services.message MWLooper dispatchMessage(MWLooper java:412)
  at com.mathworks.services.message.MWLooper.runBridge(MWLooper java:557)
  at com mathworks.services.message.MWLooper.run(MWLooper.java 526)
  at java.lang.Thread run(Unknown Source)
```

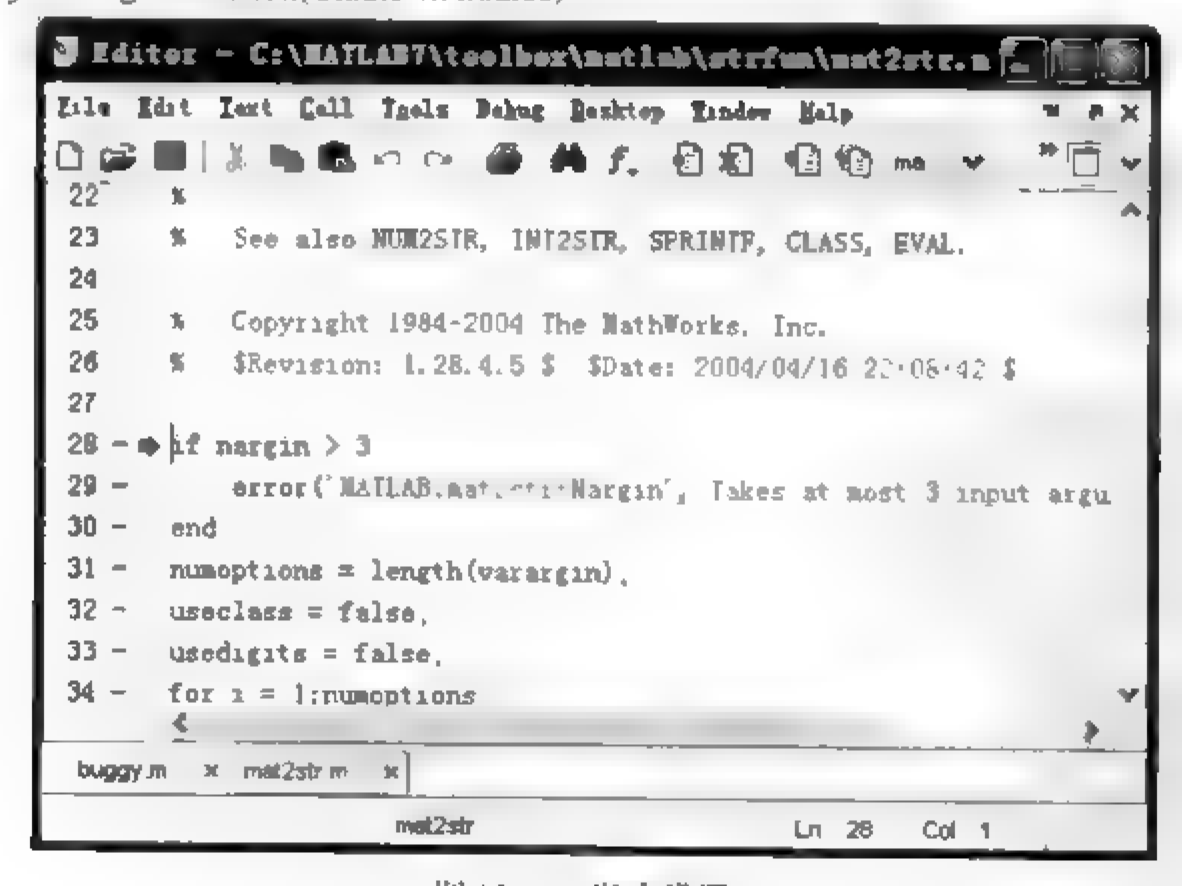


图 10-10　断点设置 2

10.5.2 程序的优化

MATLAB 7.0 语言将绝大多数的操作都集成化到功能强大的函数中，而 MATLAB 7.0 的很多运算在以矩阵为单位进行计算时都十分方便，同时，这些方面也正是如何提高 MATLAB 7.0 程序效率的关键所在。

1. 程序优劣的分析

在 MATLAB 7.0 语言中，使用 profile 函数以及计时函数 tic 和 toc 来分析程序中各个部分的耗时情况，从而帮助用户找出程序中需要改进的地方。其中 profile 在计算相对耗时以及查找文件执行过程中瓶颈问题时更为有效，而 tic 和 toc 函数在计算绝对耗时时更为有效。下面对它们的用法予以介绍。

(1) profile 函数

profile 函数可以查找 M 文件的调用记录，并且标记出哪些行相对来说占用了最多的时间。例如，如果一行或是一个函数的调用占用了一个给定 M 文件的 80%的时间，那么这一行或者是这个函数的调用所花费的时间将会对整体执行速度产生最显著的影响。此时，用户可以考虑重新编写代码以消除这些影响运行速度的代码行；也可以考虑将这些代码中的数据处理量减小到最小。其调用格式如下。

- profile on 命令开始记录 M 文件的调用，并清除以前的记录。
- profile on -detail level 命令开始记录 M 文件的调用，清除以前的记录，并标识出用户想要进行时间监控的函数。使用表 10-8 中的字符串做为 level 的值。
- profile on -history 命令开始记录 M 文件的调用，清除以前的记录，并记录确定序列的函数调用。profile 函数最多可以记录 10000 条函数的调用记录。
- profile off 命令中断 M 文件的调用记录。
- profile resume 命令重新开始 M 文件的调用记录，并且保存原来的记录。
- profile clear 命令清除 M 文件的调用记录。
- profile viewer 命令中断 M 文件的调用记录并将调用结果在 Profiler 窗口中显示。
- *s = profile('status')* 命令返回一个包含当前调用状态的结构。关于 *status* 的具体内容可以查看 MATLAB 7.0 的联机帮助系统。
- *s = profile('INFO')* 命令中断调用并返回一个包含记录结果的结构。关于 *INFO* 的具体内容可以查看 MATLAB 7.0 的联机帮助系统。

表 10-8 level 取值的含义

level 的取值	包含的信息
'builtin'	M 文件、 M 子文件、MEX 文件以及内置函数(如 eig)。
'mmex'	M 文件、 M 子文件和 MEX 文件，这也是默认值。

下面通过两个实例来演示 profile 函数的应用形式。

例 10-43　profile 函数的应用。

解：本例使用 profile 来记录 MATLAB 7.0 中 magic 函数的调用情况，并将生成的结果在 Profiler 窗口予以显示。编制程序如下。

```
>> profile on
>>plot(magic(35))
>>profile viewer
>>p = profile('info');
>>profsave(p,'profile_results')
>>
```

生成的 magic(35)矩阵的图形如图 10-11 所示，profile 记录的函数调用情况如图 10-12 所示。

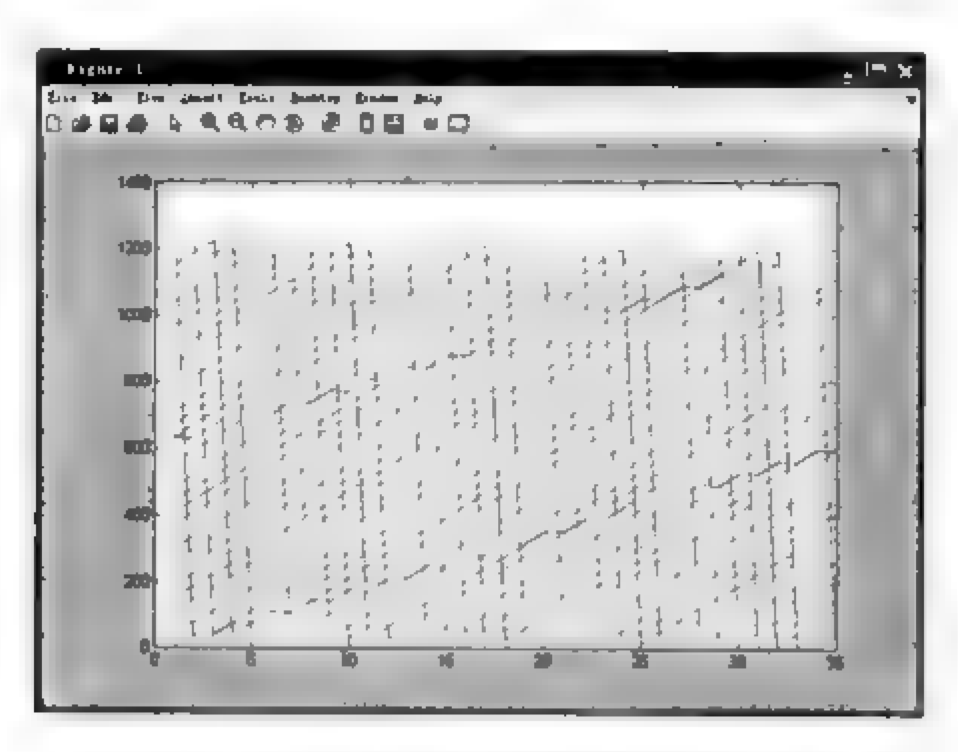


图 10-11　magic(35)矩阵的二维图

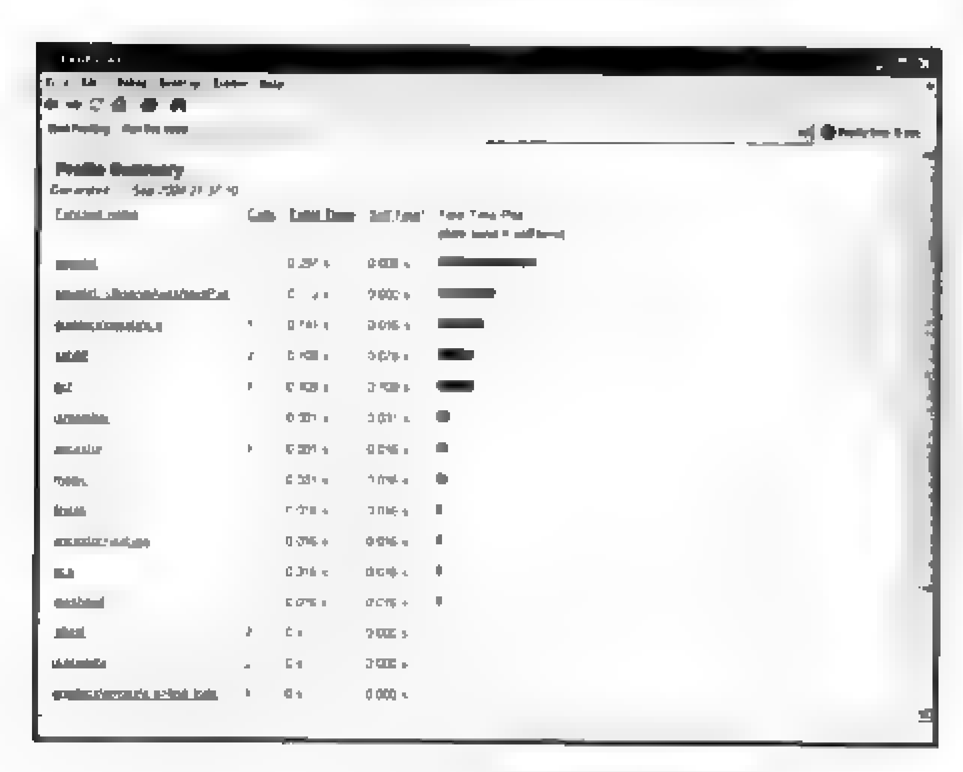


图 10-12　profile 记录的函数调用情况

例 10-44　profile 函数的应用。

解：另一种保存 profile 函数记录数据的方式是将其保存在 MAT 文件中，本例将记录数据保存在 MAT 文件中，并记录数据从内存中清除，然后再从 MAT 文件中调出记录数据。编制程序如下。

```
>> p = profile('info');
save myprofiledata p
clear p
load myprofiledata
profview(0,p)
>>
```

profile 记录的函数调用情况如图 10-13 所示。

图 10-13　profile 记录的函数调用情况

(2) tic 和 toc 函数的使用

如前所述，profile 函数在计算相对耗时以及查找文件执行过程中瓶颈问题时非常有效，而 tic 和 toc 函数在计算绝对耗时时更为有效。有关 tic 和 toc 函数的使用说明在 10.4.3 节中已经做了介绍。这里不再赘述。

2. 程序优化的技巧

(1) 程序的向量化操作

循环运算是 MATLAB 中的最大弱点，在程序设计中，应当尽量避免使用循环运算。由于 MATLAB 7.0 是一门矩阵语言，因此它是为向量和矩阵运算设计的。用户可以通过将 M 文件向量化来优化 M 文件，所谓向量化就是使用向量和矩阵运算来代替 for 循环和 while 循环。

例 10-45　比较向量运算和循环运算求解同一问题的耗时。

解：本例以 0.01 为间隔，求解从 0 到 10 内的正弦值。

首先采用循环运算，编写程序如下。

```
>> T2=cputime;
>> for t = 0:.01:10
       i = i + 1;
       y(i) = sin(t);
end
>> T=cputime-T2
T =
       1.0469
>>
```

总的耗时为 1.0469 秒。

然后再采用向量运算，编制程序如下。

```
>> T1=cputime;
>> t = 0:.01:10;
y = sin(t);
>> T=cputime-T1
T =
       0.5156
>>
```

总的耗时为 0.5156 秒，可见，采用向量运算比采用循环运算可以节省相对多的时间。

(2) 数据的预定义

使用 for 循环和 while 循环来增加数据结构的大小时，将影响系统和内存的使用。而对于未定义的变量，如果操作过程中出现越界赋值时，系统将寻找成块的内存，用来对变量进行扩充，这样极大降低了程序运行的效率。提高效率的可行办法是进行预定义，即对于可能出现变量维数不断扩大的问题，应当预先估计变量可能出现的最大维数，进行预先定义。

在 MATLAB 7.0 中，使用 zeros 和 cell 函数来进行预定义，如表 10-9 所示。

表 10-9　zeros 和 cell 函数进行预定义

矩阵类型	函　数	示　例
数值型	zeros	y = zeros(1, 100);
单元型	cell	B = cell(2, 3),B{1,3} = 1.3,B{2,2} = 'string';

下面通过一个实例来验证一下对矩阵进行预定义的效果。

例 10-46　比较对矩阵进行预定义和不进行预定义时求解同一问题的耗时。

解：首先是不进行预定义的情况，编制程序如下。

```
>> T1=cputime;
>> x = 0;
for k = 2:1000
    x(k) = x(k-1) + 5;
end
>> T=cputime-T1
T =
    1.3750
>>
```

总的耗时为 1.3750 秒。

然后是进行预定义的情况，编制程序如下：

```
> T2=cputime;
>> x = zeros(1, 1000);
for k = 2:1000
    x(k) = x(k-1) + 5;
end
>> T=cputime-T2
T =
     0.6563
>>
```

总的耗时为 0.6563 秒，可见，进行预定义比不进行预定义可以节省一定的时间。

3. 有效地使用内存

对内存的合理操作和管理将有效地提高程序运行的效率。MATLAB 7.0 提供了一系列管理内存的函数，如表 10-10 所示。

表 10-10 内存管理函数及其功能

函 数 名	函 数 功 能
whos	显示有多少内存已经被分配给工作区间的变量
pack	重新分配内存
clear	从内存中清除所有的变量
save	有选择地将变量保存至磁盘
load	从磁盘中调出由 save 命令保存的变量
quit	退出 MATLAB 7.0 环境，释放所有的内存

需要指出的是，MATLAB 7.0 本身并不具备管理系统资源的能力，所以，在进行较大规模的计算时，用户应尽可能关闭一切不必要的窗口和应用程序，以节省资源。

10.6　M 文件举例

要想熟练地掌握 MATLAB 7.0 程序的编写，需要大量的实践来实现。因为掌握 MATLAB 7.0 编程的技巧关键在于多编程，从实际中发现问题，解决问题，经过不断的积累，也有可能掌握其中的一些技巧。本节将介绍一些经典的 M 文件，用户可以仔细阅读这些文件，从中可以学习许多编程的技巧和程序优化的方法。然后，用户可以参照这些例子，再编制一些相关的程序，从而进一步提高自己编制和优化程序的能力。

1. 矩阵的乘法

本程序实现矩阵的乘法，源程序如下：

```
%% Matrix Manipulation
% This demo examines some basic matrix manipulations in MATLAB 7.0.
%
% Copyright 1984-2002 The MathWorks, Inc.
% $Revision: 5.16 $ $Date: 2002/04/15 03:35:51 $
%%
% We start by creating a magic square and assigning it to the variable A.
A = magic(3)
%%
% Here's how to add 2 to each element of A.
%
% Note that MATLAB 7.0 requires no special handling of matrix math
A+2
%%
% The apostrophe symbol denotes the complex conjugate transpose of a matrix.
%
% Here's how to take the transpose of A.
A'
%%
% The symbol * denotes multiplication of matrices.
%
% Let's create a new matrix B and multiply A by B.
B = 2*ones(3)
A*B
%%
% We can also multiply each element of A with its corresponding element of B by
% using the .* operator.
A.*B
%%
% MATLAB 7.0 has functions for nearly every type of common matrix calculation. For
% example, we can find the eigenvalues of A using the "eig" command.
```

```
eig(A)
%%
% This concludes our brief tour of some MATLAB 7.0 matrix handling capabilities.
```

2. 编制矩阵的转制函数 inverse

该函数用于求矩阵的转制，源程序如下：

```
%% Inverses of Matrices
% This demo shows how to visualize matrices as images and uses this to
% illustrate matrix inversion.
%
% Copyright 1984-2002 The MathWorks, Inc.
% $Revision. 5.14 $
%%
% This is a graphic representation of a random matrix. The RAND command creates
% the matrix, and the IMAGESC command plots an image of the matrix,
% automatically scaling the color map appropriately.
n = 100;
a = rand(n);
imagesc(a);
colormap(hot),
axis square;
%%
% This is a representation of the inverse of that matrix  While the numbers in
% the previous matrix were completely random, the elements in this matrix are
% anything BUT random. In fact, each element in this matrix ("b") depends on
% every one of the ten thousand elements in the previous matrix ("a").
b = inv(a);
imagesc(b);
axis square,

%%
% But how do we know for sure if this is really the correct inverse for the
% original matrix? Multiply the two together and see if the result is correct,
% because just as 3*(1/3) = 1, so must a*inv(a) = I, the identity matrix. The
% identity matrix (almost always designated by I) is like an enormous number
% one. It is completely made up of zeros, except for ones running along the main
% diagonal.
%%
% This is the product of the matrix with its inverse: sure enough, it has the
% distinctive look of the identity matrix, with a band of ones streaming down
% the main diagonal, surrounded by a sea of zeros.
imagesc(a*b);
axis square;
```

3. 矩阵的指数运算

本程序实现矩阵的指数运算，源程序如下：

```
%% Matrix Exponentials
% For background on the computation of matrix exponentials, see
% "Nineteen dubious ways to compute the exponential of a matrix,
% twenty-five years later," SIAM Review 45, 3-49, 2003.
% The Pseudospectra Gateway is also highly recommended.  The web site is:
% http://web.comlab.ox.ac.uk/projects/pseudospectra/
% Here are three of the 19 ways to compute the exponential of a matrix.
% Copyright 2004 The MathWorks, Inc.
%% Start with the matrix A.
A = [0 1 2; 0.5 0 1; 2 1 0]
Asave = A;
%% Scaling and squaring
% |expmdemo1| is an M-code implementation of the built-in algorithm used by
% MATLAB 7.0 for the matrix exponential. See Golub and Van Loan, Matrix
% Computations, 3rd edition, algorithm 11.3-1.
% Scale A by power of 2 so that its norm is < 1/2 .
[f,e] = log2(norm(A,'inf'));
s = max(0,e+1);
A = A/2^s,
% Pade approximation for exp(A)
X = A,
c = 1/2;
E = eye(size(A)) + c*A,
D = eye(size(A)) - c*A,
q = 6;
p = 1;
for k = 2:q
    c = c * (q-k+1) / (k*(2*q-k+1));
    X = A*X,
    cX = c*X;
    E = E + cX,
    if p
       D = D + cX;
    else
       D = D - cX;
    end
    p   ~p;
end
E = D\E;

% Undo scaling by repeated squaring
```

```
for k = 1:s, E = E*E; end
E1 = E
%% Matrix exponential via Taylor series.
% |expmdemo2| uses the classic definition for the matrix exponential. As a
% practical numerical method, this is slow and inaccurate if |norm(A)| is
% too large.
A = Asave;
% Taylor series for exp(A)
E = zeros(size(A));
F = eye(size(A));
k = 1;
while norm(E+F-E,1) > 0
    E = E + F;
    F = A*F/k;
    k = k+1;
end
E2 = E
% Matrix exponential via eigenvalues and eigenvectors.
% |expmdemo3| assumes that the matrix has a full set of eigenvectors. As a
% practical numerical method, the accuracy is determined by the condition
% of the eigenvector matrix.
A = Asave;
[V,D] = eig(A);
E = V * diag(exp(diag(D))) / V;
E3 = E
%% Compare the results.
% For this matrix, they all do equally well
E = expm(Asave);
err1 = E - E1
err2 = E - E2
err3 = E - E3
%% Taylor series fails.
% Here is a matrix where the terms in the Taylor series become very large
% before they go to zero.   Consequently, |expmdemo2| fails.
A = [-147 72; -192 93];
E1 = expmdemo1(A)
E2 = expmdemo2(A)
E3 = expmdemo3(A)

%% Eigenvalues and vectors fails.
% Here is a matrix that does not have a full set of eigenvectors.
% Consequently, |expmdemo3| fails.
A = [-1 1; 0 -1];
E1 = expmdemo1(A)
```

```
E2 = expmdemo2(A)
E3 = expmdemo3(A)
```

10.7 习　　题

1. 简述使用M文件与在MATLAB命令窗口中直接输入命令有何异同？有何优缺点？

2. 简述脚本式M文件与函数式M文件的异同。

3. 当一个函数式M文件被调用时，函数和被调用之间怎样实现数据传递？

4. 编制一个程序，计算 $y(x)=\begin{cases}-3x^2+5(x\ge 0)\\3x^2+5(x<0)\end{cases}$ 的值，其中 x 的值为-10到10之间，以0.5为步长，使用循环语句加以实现。

5. 重新编制程序计算例10.4，并采用向量的形式加以实现，并使用时间运算函数将本例的程序与例10.4的程序加以比较。

6. 试计算以下循环语句将进行多少步操作。

(1) for i=-32768:32767

(2) for j=32768:32767

(3) for k=2:4:3

(4) for m=ones(5,5)

7. 观察以下循环语句，试计算每个循环的循环次数和循环结束之后 ires 的值。

(1) ires=1;

```
while mod(ires,10)~=0
   ires=ires+1;
end
```

(2) ires=2;

```
while ires<=200
    ires=ires^2;
end
```

(3) ires=2;

```
while   ires>200
      ires=ires^2;
end
```

8. 编制程序计算如下函数：

(1) $\sinh(x)=\dfrac{e^x-e^{-x}}{2}$

(2) $\cosh(x)=\dfrac{e^x+e^{-x}}{2}$

(3) $\tanh(x)=\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$

9. 利用 random 函数编制一个新的函数 random10，该函数能够产生在[-10,10]之内的随机数，注意使 random 函数为该函数的子函数。

10. 将上例的函数加以修改，使得产生的随机数在[low,high]之间，其中 low 和 high 为两个调用参数。

第11章　文件和数据的导入与导出

MATLAB 7.0 使用多种文件格式打开和保存数据，有的文件格式是为 MATLAB 7.0 定制的，有的是业界标准的文件格式，还有一些是为其他应用程序定制的文件格式。用来保存和打开数据文件的技术包括 GUI 以及命令窗口命令。

与大多数现代应用程序一样，MATLAB 7.0 把当前目录作为数据文件和 M 文件的默认目录。目录管理工具和改变当前目录都是通过 GUI 和命令窗口函数来实现的。

11.1　本机数据文件

11.1.1　文件的存储

可以用 save 变量将 MATLAB 7.0 工作区中的变量存储为 MATLAB 7.0 本机格式。例如:

```
>>save
```

这条命令将 MATLAB 7.0 工作区中的所有变量以 MATLAB 二进制的格式存储在当前目录下的 matlab.mat 文件中。这些本机二进制 MAT 文件都是完整的双精度数，并且还保留了变量名字。MAT 文件不是与平台无关的，但它完全是跨平台使用的。在一个平台下保存的变量可以在其他平台上打开，而不需要任何特殊的处理。

用 save 也可以用来保存特定的变量，例如:

```
>>save var1 var2 var3
```

这条命令只将变量 var1 var2 和 var3 保存在 matlab.mat 文件中。这个文件名还可以声明为 save 的第一个参数，例如:

```
>>save filename var1 var2 var3
```

这条命令将 var1 var2 和 var3 保存在名为 matlab.mat 的文件中。

利用命令-函数的两重性，上边这种命令形式还可以写成函数形式，例如:

```
>>save(' filename', ' ar1', ' var2', 'var3')
```

在文件名是用一个 MATLAB 7.0 字符串保存的情况下，这种特殊的格式是非常有用的，例如：

```
>>fname='myfile'
>>save (fname, 'var1', 'var2', 'var3')
```

这里，指定的变量是被保存在一个名为 myfile.mat 的文件中。

除了上述这些简单的形式之外，svae 命令还可以将变量保存为 ASCⅡ文本格式，并且可以用来数据附加到一个已经存在的文件的后边。关于这些特性的帮助信息，请参看联机帮助文档。

11.1.2　文件的打开

save 命令的补函数就是 load 命令。这个命令打开用 save 命令创建的数据文件或者打开适用于 save 命令的数据文件。例如:

```
>>load
```

这条命令载入在当前目录下或者在 MATLAB 7.0 的搜索路径中找到的第一个 matlab.mat 中的所有变量。原来存在的 matlab.mat 中的变量也被保存在工作区中，并且它们会覆盖在任何工作区中已经存在的同名变量。

为了从一个 MAT 文件中载入特定的变量，用户必须将文件名和变量列表包含进去，例如:

```
>>load filename var1,var2,var3
>>load ('filename', 'var1', 'var2', 'var3,')
```

这里，filename.mat 被打开，然后变量 var1、var2 和 var3 被载入工作区。第二条语句展示了 load 命令的函数形式，这种形式允许用字符串来声明数据文件。尽管这里没有显示出来，但要知道 filename 可以包括一个完整的或者部分的路径，这样就将 load 限定在一个指定的目录内去寻找所需要的数据文件。

前面的示例曾提供了一个打开一组编了号的数据文件的方法，例如 mydata1.mat 和 mydata2.mat 等，例如：

```
for   i=1:N
        fname=sprintf('mydata%d',i);
        load (fname)
end
```

这段代码用 sprintf 来在一个 for 循环中来生成文件名字符串，这样的一组数据文件都可以载入到工作区中。

当用户不想将工作区中的变量覆盖时候，命令 load 可以写成函数的形式并且返回一个输出参数。例如:

```
>>vnew= load ('filename', 'var1', 'var2');
```

这条命令打开 filename.mat 并且将变量 var1 和 var2 载入到一个名为 vnew 的结构变量中，这个结构变量有两个域，一个是 var1，一个是 var2。也就是说，vnew.var1=var1，vnew.var2=var2。

load 命令还可以打开 ASCII 文本文件。特别是当数据文件是由 MATLAB 7.0 的注释行和一行一行用空格隔开的值组成的时候，例如：

```
>>load filename.ext
```

打开文件 filename.ext，然后将数据载入到一个名为 fielname 的双精度数据组中。更详细的信息可以参见 MATLAB 7.0 中关于 load 命令的帮助系统。

为了查出一个数据文件是否存在，以及它包含哪些变量，就是使用 MATLAB 7.0 命令 exist 和 whos，例如：

```
>>exist ('matlab.mat' , 'file')
```

如果文件不存在，就返回 0，如果存在，就返回 2。

```
>>who-file Matlab.mat
```

这条命令返回 matlab.mat 文件中包含的变量，其显示格式为标准 whos 在命令窗口中的显示格式，作为替代：

```
>>w=whos('-file', 'matlab.mat')
W=
3×1 struct array with fields
   Name
Size
Bytes
Class
```

这条命令返回一个结构数组，其域名表示的是 whos 显示的列名。用这种方法，可以将变量名，大小，内存和类保存在变量中。

最后，数据文件可以用命令窗口命令 delete 进行删除，例如：

```
>>delete filename.ext
```

这条命令删除了 filename.ext 的文件。

在 MATLAB 7.0 中，数据文件管理函数可以通过工作区浏览器以及导入向导进行访问。Workspace 浏览器可以通过选择 MATLAB 7.0 桌面的 View 菜单中的 Workspace 来浏览。导入向导可以通过选择 File 菜单中的 Import Data…，或者在命令窗口中输入 uiimport 得到。它是一个多用途的 GUI，让用户很容易地用不同的格式，而不仅仅以 MATLAB 7.0 的本地 MAT 文件格式载入数据。

11.2　数据导入和导出

除了 MATLAB 7.0 的本地 MAT 文件格式以及传统的 ASCII 文本格式之外，MATLAB 7.0 还支持多个工业标准格式和一些用户文件格式。有的格式被限定为只读，而有的格式被限定为只写。有的文件格式被限定为图形文件格式，而有的格式被限定为多媒体或者电子表格文件格式。这些数据导入和导出函数及其功能使得 MATLAB 7.0 能够和其他的程序交换数据。

可以使用 Figure 窗口中 File 菜单中的 Save 选项将 Figure 窗口图形保存为本地的 MATLAB 7.0 的 FIG 文件格式。另外，可以通过选择 Figure 窗口中 File 菜单中 Export...项将 Figure 窗口图形导出为多种文件格式。命令窗口函数 saveas 提供了这种基于 GUI 的方法的一种替代方法。关于 saveas 的详细信息用户可以参见 MATLAB 7.0 的帮助系统。

在 MATLAB 7.0 中提供的专门针对数据导入和导出的函数包括表 11-1 中所列的这些函数。

表 11-1　数据的导入和导出函数及其功能

函 数 名	功 能 描 述
dlmread	把 ASCII 码中的数据输入矩阵
dlmwrite	把矩阵写入 ASCII 文件
textread	把文件读入格式化的文本
wklread	从电子表格文件读入
wklwrite	写入电子表格文件
xlsread	从电子表格文件读入
aviread	从电影文件读入
imread	从图形文件读入
imwrite	写入图形文件
auread	从 sun 声音文件读入
auwrite	写入 sun 声音文件
wavread	从 Microsoft 声音文件读入
wavwrite	写入 Microsoft 声音文件
hdf	MATLAB 7.0－HDF 网关函数

表中这些函数的帮助文档都提供了关于它们用法的信息。函数 imread 和 imwrite 还特别提供了对多种文件格式的支持。它们支持的文件格式包括 JPEG，BMP，PNG，HDF，PCX 以及 XWD 等。关于文件格式的详细信息用户可以参见 fileformats 的帮助信息。

11.3　低级文件 I/O

因为存在着如此多的文件格式，因此 MATLAB 7.0 提供了低级文件 I/O 函数来读取或写入任何可能的二进制或者格式化的 ASCII 文件。这些函数与 ANSIC 编程语言中的那些函数非常类似，但是函数名并不一定完全一致。实际上，上面描述的这些专用的文件 I/O 命令中的很多函数都在内部使用了这些 C 语言中的命令。表 11-2 列出了 MATLAB 7.0 中的低级文件 I/O 文件。

表 11-2　MATLAB 7.0 中的低级文件 I/O 文件

函数名	功能描述
fopen	打开文件
fclose	关闭文件
fread	读取一个二进制文件的全部或是部分
fwrite	将数组写入二进制文件
fscanf	从文件中读取格式化数据
fprintf	将格式化数据写入文件
fgetl	从文件读取行，并删除换行符
fgets	从文件中读取行，并保留换行符
sprintf	将格式化数据写入字符串
sscanf	在格式控制下读取字符串
ferror	获取文件 I/O 状态的信息
feof	检测是否到了文件的结尾
fseek	设置文件定位指针
ftell	获取文件定位指针的位置
frewind	将文件定位指针设置在文件的开头

关于各个函数的具体使用方法，用户可以参考 MATLAB 7.0 的帮助系统，在此不再赘述。

11.4　习　　题

1. 讨论二进制和带格式 I/O 文件的区别，并通过帮助系统学习表 11-2 中函数的使用方法。

2. 编制一个程序，该程序可以从用户指定的输入数据文件中读入任意数目的实型数值，并将读入的数据进行四舍五入，并将处理过的数据读出到一个另外的文件。当读入文件不存在时，程序将给出提示。如果读出文件已经存在，程序将提示输出另外的文件。

3. 判断下面程序是否正确，如果不正确，请指出错误的类型。

(1)
```
a=2*pi;
b=6;
c='hello';
fprintf(fid,'%s %d %g\n',a,b,c);
```

(2)
```
data1=1:20;
data2=1:20;
fid=fopen('xxx','w+');
fwrite(fid,data1);
fprintf(fid,'%g\n',data2);
```

4. 判断下列语句的输出结果。

(1) sprintf('%0.5g',(1+sqrt(5))/2)

(2) sprintf('%0.5g',1/eps)

(3) sprintf('%15.5f',1/eps)

(4) sprintf('%d',round(pi))

(5) sprintf('%s','hello')

(6) sprintf('The array is %dx%d.',2,3)

(7) sprintf('\n')

第12章 图 形 处 理

与数值计算和符号计算相比，图形的可视化技术是数学计算人员所追求的更高级的一种技术，因为对于数值计算和符号计算来说，不管计算的结果是多么的准确，人们往往无法直接从大量的数据和符号中体会它们的具体含义。而图形处理技术则给人们提供了一种更直接的表达方式，可以使人们更直接、更清楚地了解事物的结果和本质。MATLAB 7.0 语言除了有强大的矩阵处理功能之外，它的绘图功能也是相当强大的。本章将主要介绍 MATLAB 7.0 的图形处理功能，包括基本的绘图命令、图形的简单控制、图形窗口的编辑以及图形的高级控制等。

12.1 基本的绘图命令

MATLAB 7.0 语言在绘图方面的功能非常全面，可以很方便地绘制二维、三维甚至多维图形。本节将主要介绍使用 MATLAB 7.0 语言进行基本的图形绘制，并介绍一些简单的图形操作命令。

12.1.1 图形窗口简介

MATLAB 7.0 语言的绘图函数和工具将所绘制的图形在图形窗口中显示出来，该窗口将与 MATLAB 7.0 的主窗口隔离出来。如图 12-1 标识了图形窗口的主要部分。

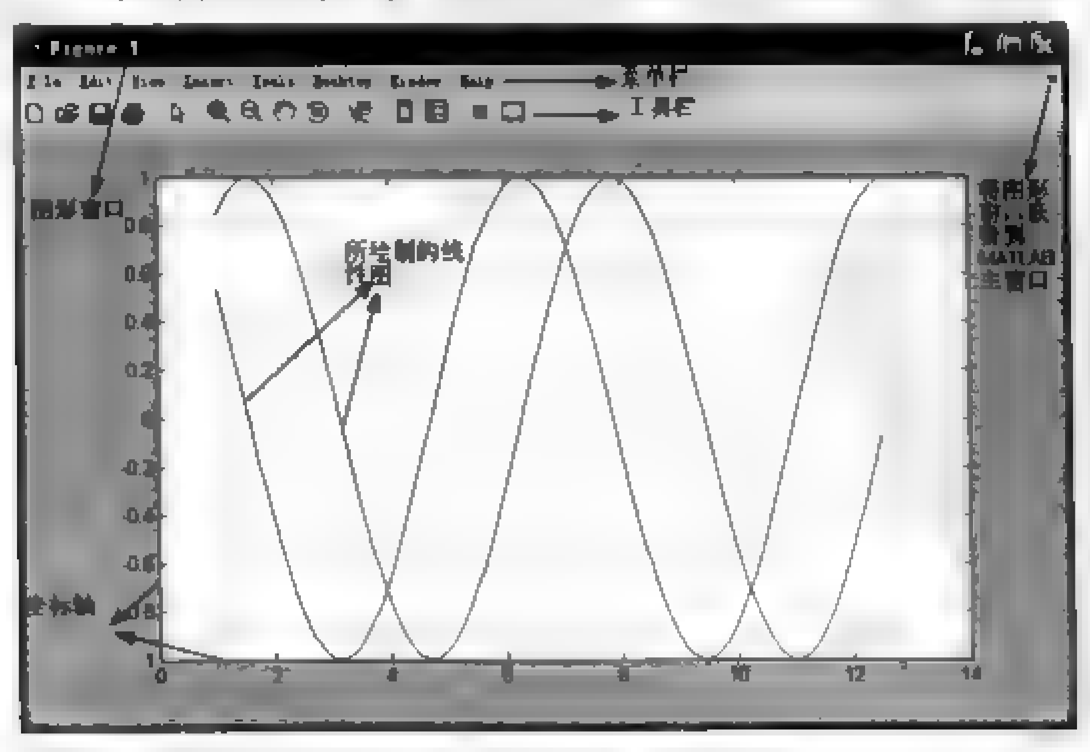


图 12-1 图形窗口

在默认情况下，MATLAB 7.0 语言使用不同的线型和颜色来区分图形中的数据。当然，用户也可以修改图形中各元素的线型和颜色，或是在图形中增加注释，从而更好地表达图形。图形窗口默认的工具栏为一些常用的命令的快捷方式，下面简单介绍一下工具栏中各个图标的功能，如图 12-2 所示。

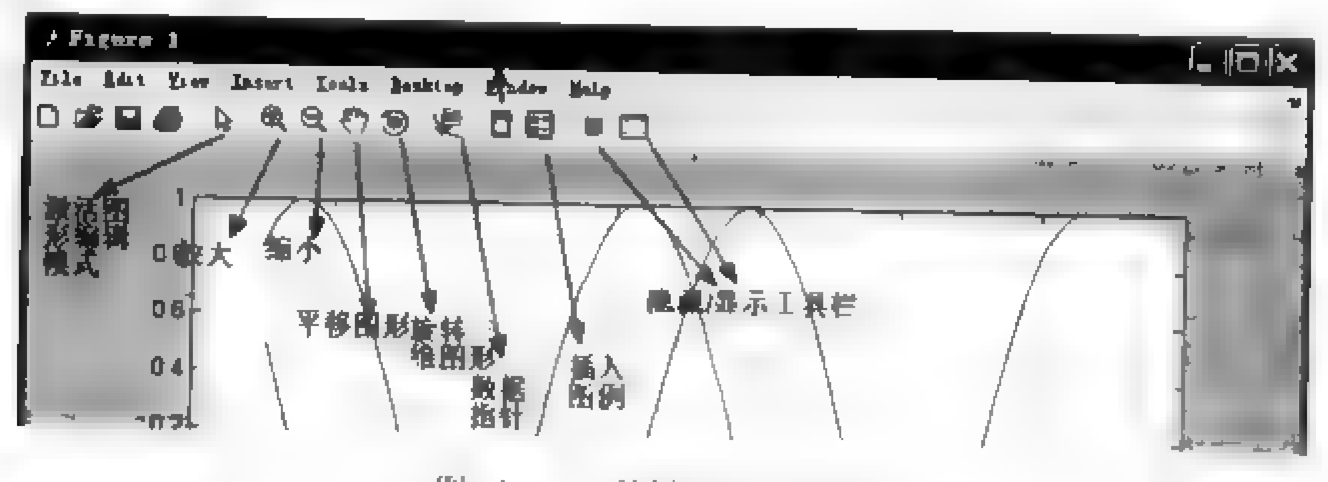


图 12-2 工具栏图标功能简介

12.1.2 基本的绘图操作

MATLAB 7.0 语言提供了一系列的函数来将向量数据以线性图的形式表示出来，以及一些注释和打印图形的函数。表 12-1 列出了一些绘制基本线性图的函数。这些函数与坐标轴的比例关系各不相同，但它们的输入形式都是向量或是矩阵，并自动按比例调整坐标轴的比例来适应这些数据。

表 12-1 绘制基本线性图的函数表

函 数 名	功 能 描 述
plot	在 x 轴和 y 轴都按线性比例绘制二维图形
plot3	在 x 轴、y 轴和 z 轴都按线性比例绘制三维图形
loglog	在 x 轴和 y 轴按对数比例绘制二维图形
semilogx	在 x 轴按对数比例，y 轴按线性比例绘制二维图形
semilogy	在 y 轴按对数比例，x 轴按线性比例绘制二维图形
plotyy	绘制双 y 轴图形

1. 绘图步骤

创建一个基本图形的过程如表 12-2 所示，该表显示了一些典型的步骤以及每个步骤的一些典型实例。如果用户只是进行分析操作，那么用户只需要将各种数据在图形中显示出来，在这种情况下，只需要进行第 1 步和第 3 步操作即可；如果用户在创建描述性的详图，那么就需要进行诸如图形定位、设置线型和颜色、增加注释以及其他的操作。

表 12-2　基本的绘图步骤

步　　骤	典 型 代 码
1. 准备绘图数据	x = 0:0.2:12;y1 = bessel(1,x); y2 = bessel(2,x);y3 = bessel(3,x);
2. 选择一个窗口并在窗口中给图形定位	figure(1) subplot(2,2,1)
3. 调用基本的绘图函数	h = plot(x,y1,x,y2,x,y3);
4. 选择线型和标记特性	set(h,'LineWidth',2,{'LineStyle'},{'--';':';'-.'}) set(h,{'Color'},{'r';'g';'b'})
5. 设置坐标轴的极限值、标记符号和网格线	axis([0 12 -0.5 1]) grid on
6. 使用坐标轴标签、图例和文本对图形进行注释	xlabel('Time');ylabel('Amplitude') legend(h,'First','Second','Third') title('Bessel Functions');[y,ix] = min(y1); text(x(ix),y,'First Min \rightarrow',... 'HorizontalAlignment','right')
7. 输出图形	print -depsc -tiff -r200 myplot

2. 绘制二维曲线图

二维曲线图在 MATLAB 7.0 中的绘制是最为简便的。如果将 X 轴和 Y 轴的数据分别保存在两个向量中，同时向量的长度完全相等，那么可以直接调用函数进行二维图形的绘制。在 MATLAB 7.0 中，使用 plot 函数进行二维曲线图的绘制，其使用格式如下：

- $plot(y)$ 命令依据 y 每列的标志绘制出的 y 每一列。如果 y 属于复平面，那么 $plot(y)$ 等价于 $plot(real(y),imag(y))$，即以 $real(y)$ 为横坐标，以 $imag(y)$ 为纵坐标来绘制二维图形。当输入变量多于一个时，虚部都将会被忽略。
- $plot(x,y)$ 命令绘制向量 y 相对向量 x 的图形。如果 x 或者 y 为矩阵的形式，那么绘制的向量则对应于矩阵中的行或者列；如果 x 是一个标量而 y 为一个向量，那么 $length(y)$ 形成不连续的点被 MATLAB 7.0 绘制出来。
- $plot(x,y,s)$ 命令可以用来绘制不同线型、标识和颜色的图形，其中 s 为一个字符串，它们的意义如表 12-3 所示。例如：$plot(x,y,'c+:')$ 所绘制的曲线在每个数据点都由“+”组成。而 $plot(x,y,'bd')$ 所绘制的曲线在每个数据点都由蓝色的棱形组成，并且在这些点之间没有线相连。
- $plot(x1,y1,s1,x2,y2,s2,x3,y3,s3...)$ 命令可以将多个图形放置在一个图形框里。其中 $x's$ 和 $y's$ 为向量或矩阵，$s's$ 为字符串。例如 $plot(x,y,'y-',x,y,'go')$ 将 x-y 曲线在图形框中显示两次，但每次所用的线型和颜色不同。

例 12-1　使用 $plot(x,y)$ 命令绘制简单的二维图形，首先以增量 0.01 创建一个在[0, 2]范围内的向量 x，然后在该范围内求向量 x 的正弦值，生成向量 y 。MATLAB 7.0 将把向量 x 定位在 X 轴上，把向量 y 定位在 Y 轴上。

解：在命令窗口中输入代码如下。

```
>> x= 0:pi/100:2*pi;
>> y=sin(x);
>> plot(x,y)
>>
```

所得图形如图 12-3 所示。

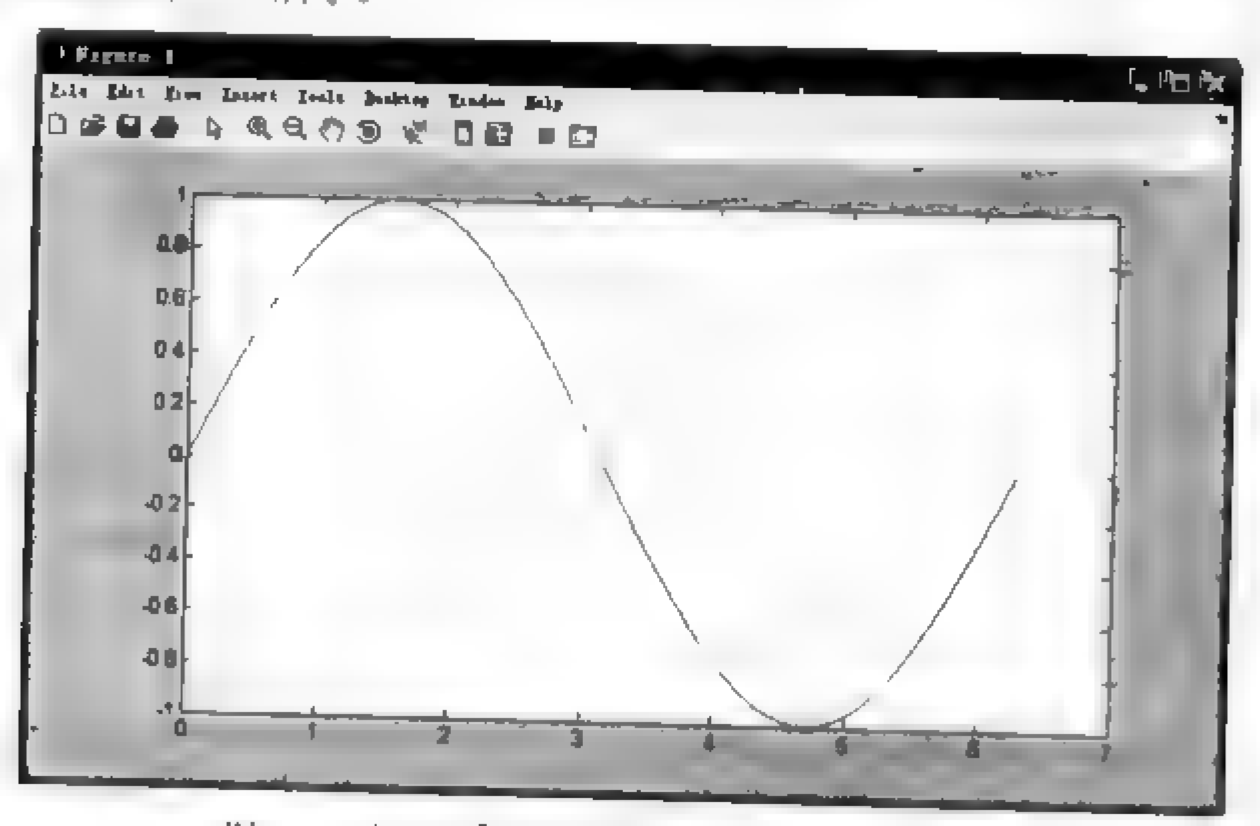


图 12-3 使用 $plot(x, y)$ 命令绘制简单的二维图形

用户也可以只调用一次命令来绘制多线图，MATLAB 7.0 将自动使用预定义的颜色来分配给不同的曲线，从而有效地区分各条曲线。

例 12-2 使用 plot 函数绘制二维多线图形。首先以增量 0.01 创建一个在[0, 2]范围内的向量 x，然后在该范围内求向量 x 的正弦值，生成向量 $y1$；然后求该范围内向量 $x-0.25$ 的正弦值，生成向量 $y2$；再求该范围内向量 $x-0.5$ 的正弦值，生成向量 $y3$。MATLAB 7.0 将把向量 x 定位在 X 轴上，把向量 $y1$、$y2$ 和 $y3$ 定位在 Y 轴上。

解：在命令窗口中输入代码如下。

```
>> x= 0:pi/100:2*pi;
>> y1=sin(x);
>> y2 = sin(x-0.25);
>> y3 = sin(x-0.5);
>> plot(x,y1,x,y2,x,y3)
>>
```

所得图形如图 12-4 所示。从中可以看出，MATLAB 7.0 自动给 3 条曲线赋予了蓝色、绿色和红色。

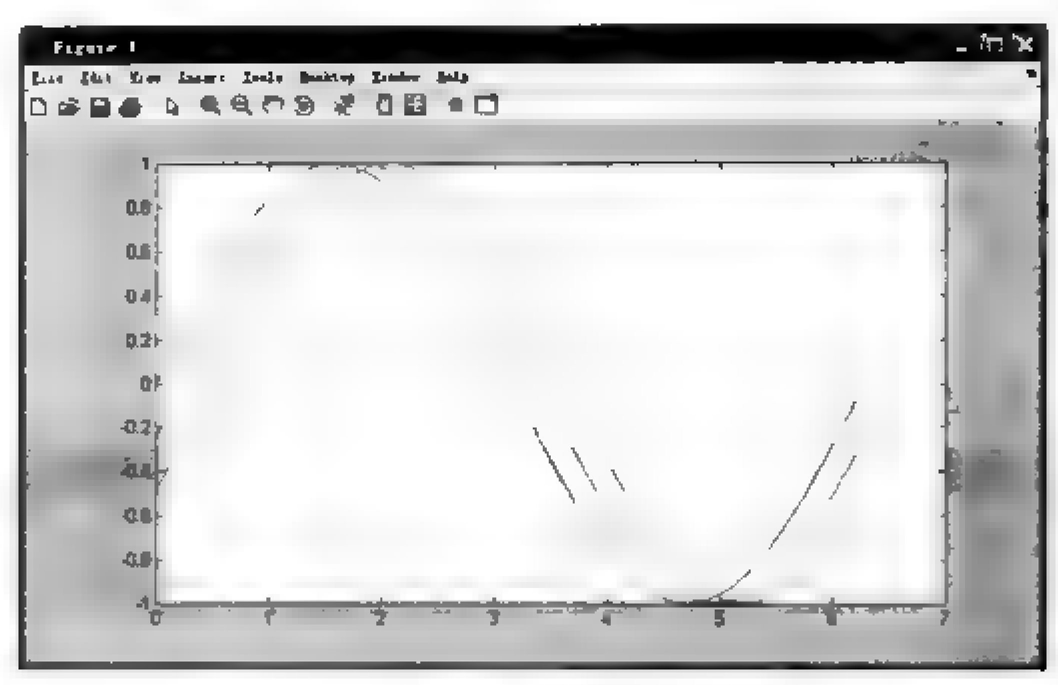


图 12-4　使用 plot 函数绘制多线图

3. 曲线的色彩、线型和数据点型

MATLAB 7.0 提供了丰富的个性化绘图工具。基本的绘图命令都支持使用字符串来给不同的曲线定义不同的线型、数据点型和色彩，如 $plot(x,y,s)$ 命令中的 s 。表 12-3 介绍了 MATLAB 7.0 中不同颜色、数据点型和线型的含义。

表 12-3　曲线的色彩、线型和数据点型参数定义

颜色符号	含义	数据点型	含义	线型	含义
b	蓝色	.	点	-	实线
g	绿色	x	X 符号	:	点线
r	红色	+	+号	-.	点划线
c	蓝绿色	h	六角星形	—	虚线
m	紫红色	*	星号	(空白)	不画线
y	黄色	s	方形		
k	黑色	d	菱形		
		v	下三角		
		^	上三角		
		<	左三角		
		>	右三角		
		p	正五边形		
		O	圆圈		
		(空白)	默认点型		

下面将通过实例对这些操作进行简要的介绍。

例 12-3　同例 12-2，使用 $plot(x1,y1,s1,x2,y2,s2,x3,y3,s3...)$ 命令绘制用户所选线型、数据点型和色彩的二维图形。

解：在命令窗口中输入代码如下。

```
>> x= 0:pi/100:2*pi;
>> y1=sin(x);
>> y2 = sin(x-0 25),
>> y3 = sin(x-0.5);
>> plot(x,y,'-.b', x,y2,'--r*', x,y3,'-.gh')
>>
```

所得图形如图 12-5 所示。从图中可以看出，本程序段分别使用了实线、虚线和点线 3 种线型，使用了点、星号和六角星号 3 种数据典型，并使用了蓝色、红色和绿色 3 种颜色。

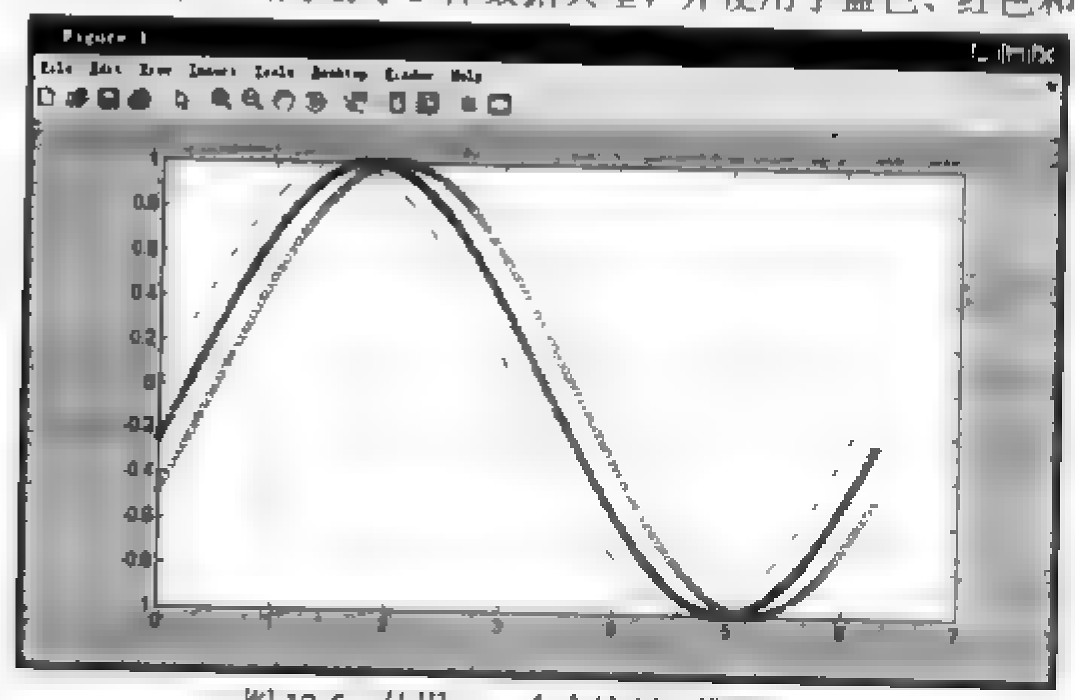


图 12-5　使用 *plot* 命令绘制二维图形。

4. 定义线的颜色和宽度

用户还可以通过指定曲线各个特性的值来控制曲线的样式。这些特性主要有：

- LineWidth ——以点为单位的宽度。
- MarkerEdgeColor——数据点型或是其边界的颜色(圆、棱形、六角星形和星号等)。
- MarkerFaceColor——数据点型的填充色。

例 12-4　定义线的颜色和尺寸。

解：在命令窗口中输入如下语句，并按 Enter 键确认。

```
>>x = -pi:pi/10:pi,
>>y = tan(sin(x)) - sin(tan(x));
>>plot(x,y,'--rs','LineWidth',2,..
                'MarkerEdgeColor','k',..
                'MarkerFaceColor','g',
                'MarkerSize',10)
>>
```

生成的曲线如图 12-6 所示。该图所绘制的曲线为红色，使用线型为虚线，数据点型为方形，曲线的宽度为 4 个点的宽度，数据点型的边界颜色为黑色，填充颜色为绿色，数据

点的宽度为 10 个点的宽度。

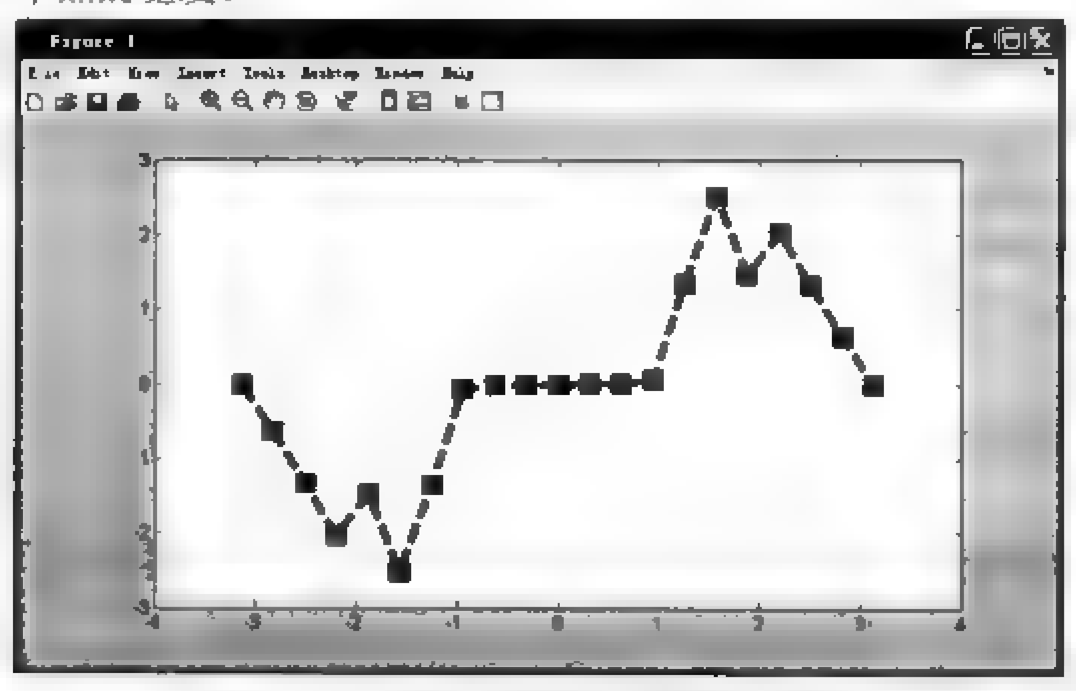


图 12-6　曲线及其数据点型的颜色和宽度

5. 图形的多次叠放

用户常会遇到需要绘制分段函数的图形，或者在一张图纸中需要绘制多条曲线，这就要求将多条曲线叠放在一起，在 MATLAB 7.0 语言中提供了 hold 函数来实现该功能。它的调用形式有：

- hold 命令并不改变坐标轴的自定义范围的性质。
- hold on 命令保持当前图形，并且将此后绘制的所有图形添加到当前的图形窗口中，如果新的曲线所对应的坐标极限值与原图不一致，系统将自动进行调整。
- hold off 命令返回 plot 命令所默认的形式，并且在绘图之前重新设置坐标系的属性。
- hold 命令在 hold on 和 hold off 状态之间进行切换。
- hold all 命令保留当前的颜色和线型，这样在绘制随后的图形时就使用当前的颜色和线型。
- *hold*(*ax*,...) 命令将同时保留坐标轴的信息。

例 12-5　使用 hold 函数绘制分段函数。

解：分段函数如下

x 为 0 到 1 时，$y = x$；

x 为 1 到 2 时，$y = 0.5x^4 + 0.5$；

x 为 2 到 5 时，$y = -x^2 + 9x - 5.5$。

在命令窗口输入如下程序，并按 Enter 键确认输入，运行程序，运行结果如图 12-7 所示。

```
>> % 该程序用于绘制分段函数图形
>> %  x 为 0 到 1 时，  y=x
>> x=0:0.01:1;
>> y=x;
>> plot(x,y)
>> % HOLD ON 命令保持当前图形，将此后绘制的所有图形添加到当前的图形窗口中
>> %新的曲线所对应的坐标极限值与原图不一致，系统将自动进行调整
```

```
>> hold on
>>
```

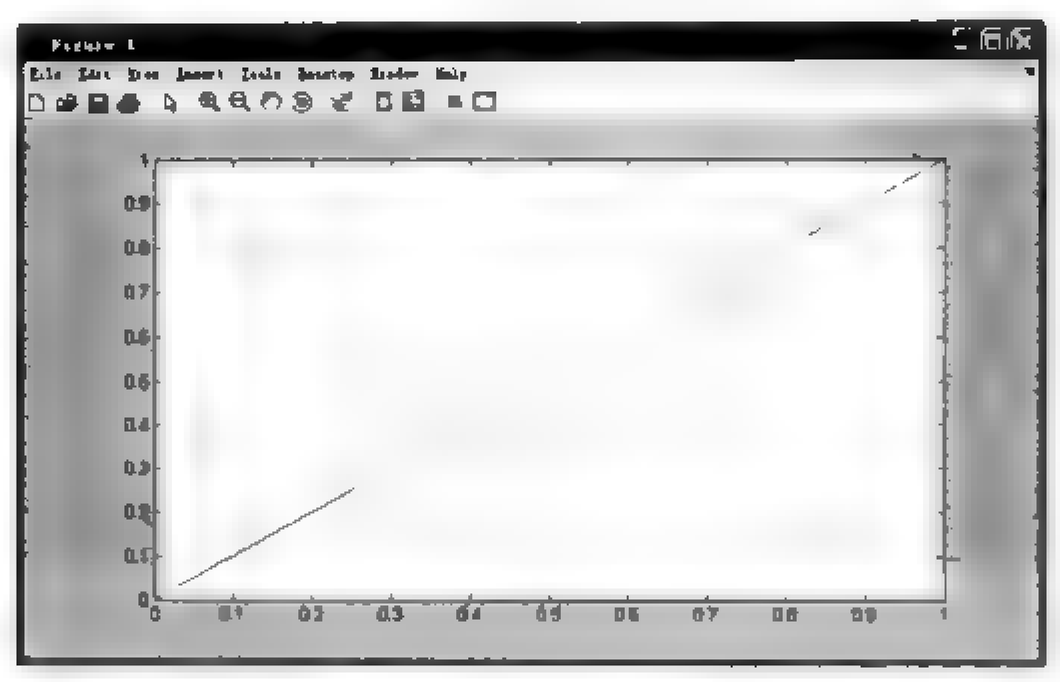


图 12-7　分段函数第 1 段

继续在命令窗口输入如下程序，并按 Enter 键确认输入，运行结果如图 12-8 所示：

```
>> % x 为 1 到 2 时，y=0.5*x.^4+0.5
>> x=1:0.01:2;
>> y=0.5*x.^4+0.5;
>> plot(x,y)
>>
```

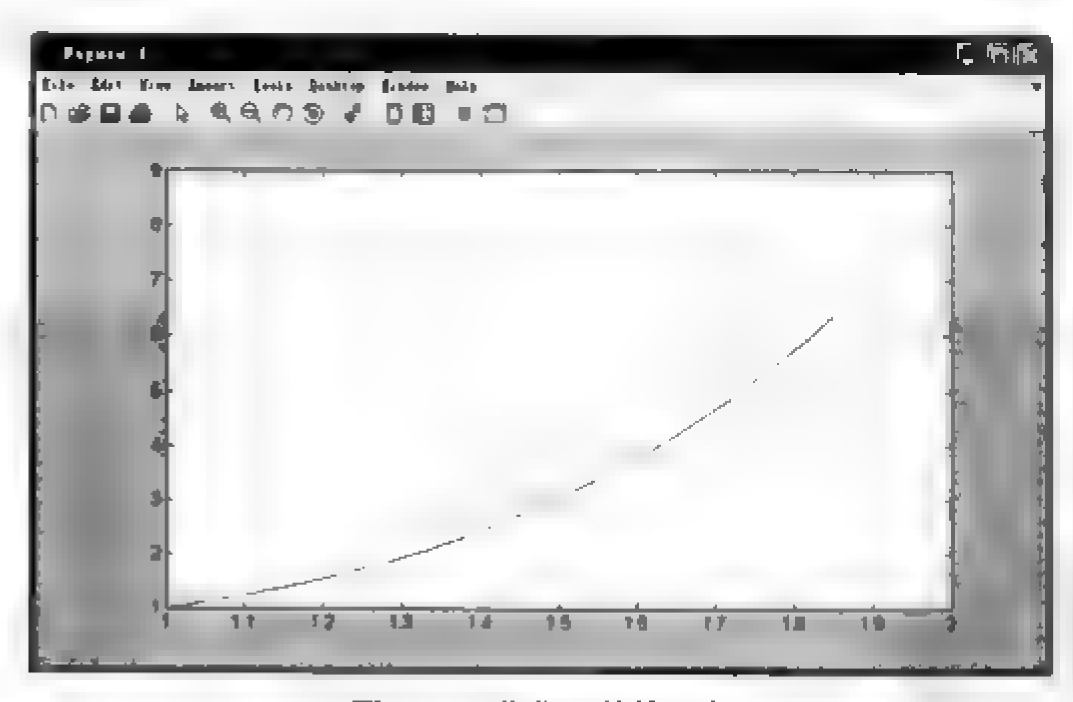


图 12-8　分段函数第 2 段

继续在命令窗口输入如下程序，并按 Enter 键确认输入，运行结果如图 12-9 所示：

```
>> % x 为 2 到 5 时，y=-x.^2+9*x-5.5
>> x=2:0.01:5;
>> y=-x.^2+9*x-5.5;
>> plot(x,y)
>>
```

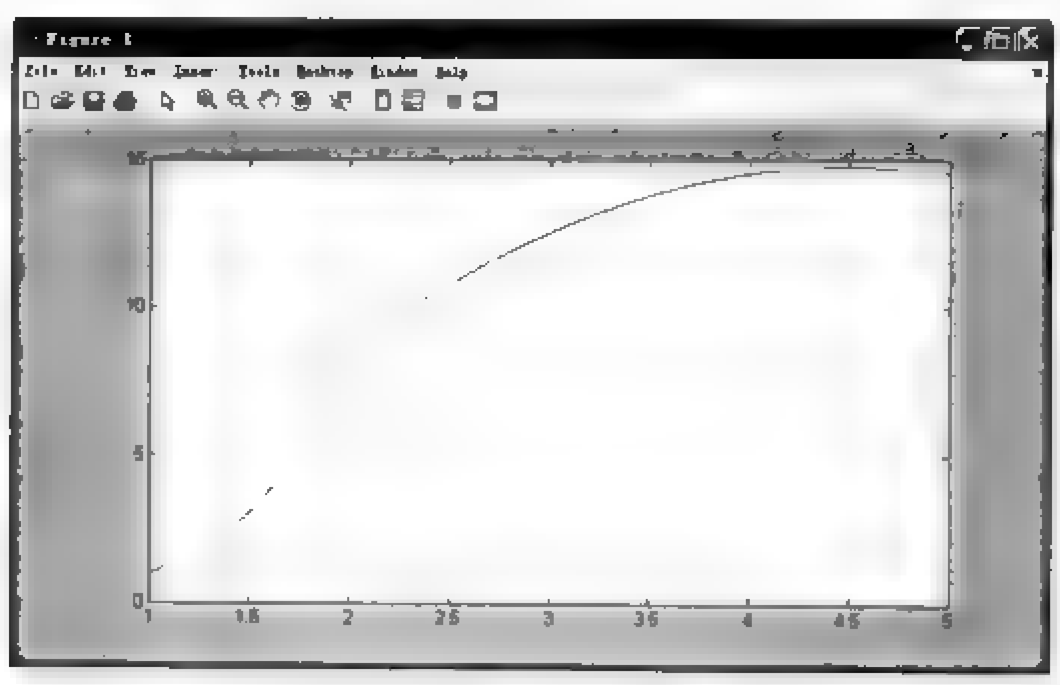


图 12-9　分段函数第 3 段

例 12-6　使用 hold 函数绘制 3 个三角函数。

解： 在命令窗口输入如下程序，并按 Enter 键确认输入，运行结果如图 12-10 所示。

```
>> % 该程序用于绘制分段函数图形
>> % x 从 0 到 4*pi
>> x=0:pi/15:4*pi;
>> y=sin(x);
>> S1='--b*';
>> plot(x,y,S1)
>> hold on
>> y=cos(x);
>> S2='-.rd';
>> plot(x,y,S2)
>>
```

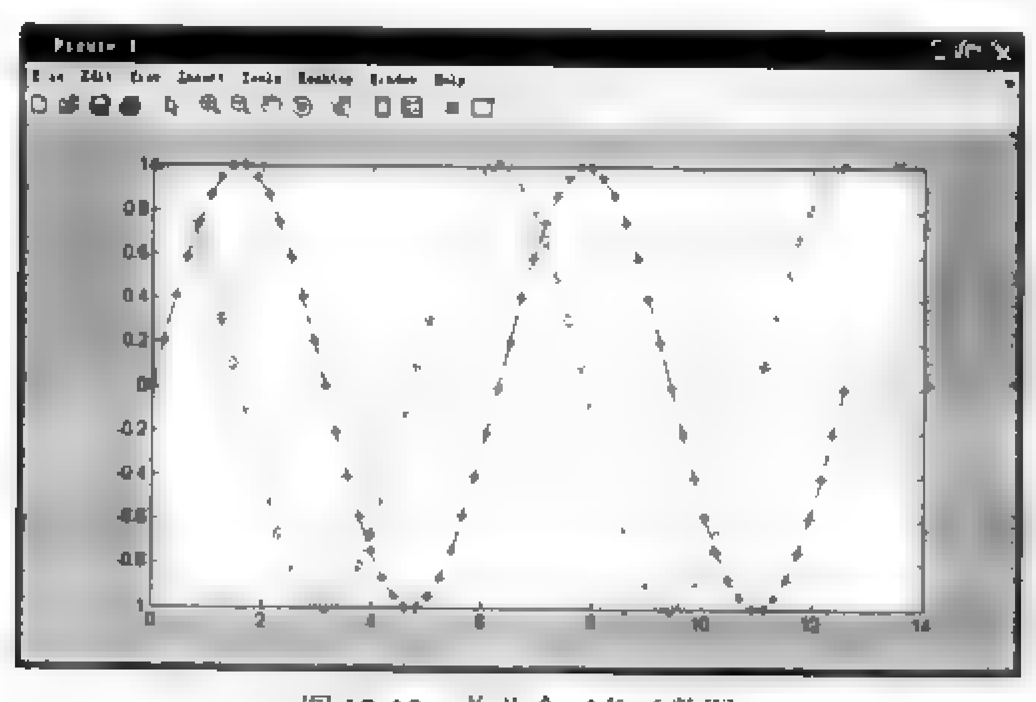


图 12-10　前两个三角函数图

继续在命令窗口输入如下程序，并按 Enter 键确认输入，运行结果如图 12-11 所示。

```
>> y=cos(x).^3+sin(x).^3;
>> S3='-kx';
>> plot(x,y,S3)
>>
```

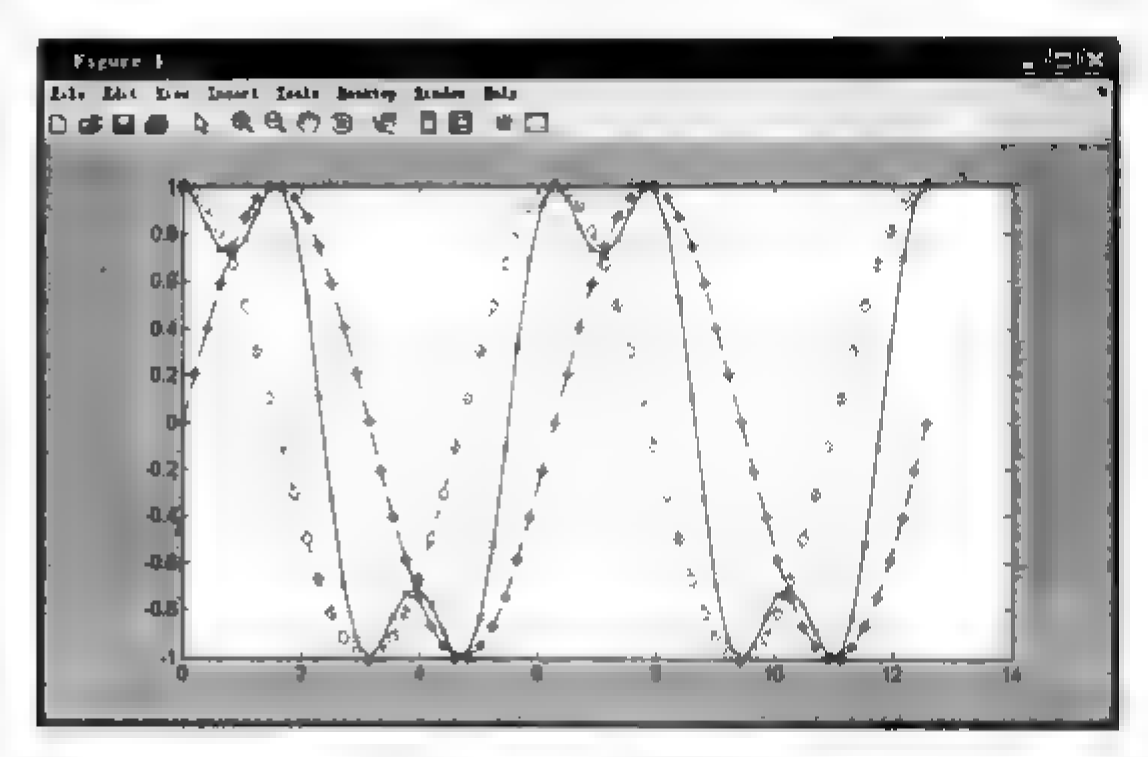


图 12-11 3 个三角函数图

6. 仅绘制二维图形的数据点

如果用户只需要将数据点绘制出来，而不需使用线型将各个点联系，那么，只需在定义线型、数据点型和色彩的字符串中去掉线型即可。

例 12-7 仅绘制数据点的二维图形。

解：在命令窗口输入如下程序，并按 Enter 键确认输入，运行程序，运行结果如图 12-12 所示。

```
>>本图所绘的图形绘制曲线 y = exp(2*cos(x))
>> x 的范围是从 0 到 4 pi.
>>y 轴的范围从 0 到 8，x 轴的范围从 0 到 14
>>在每一个数据点有一个红色的加号，但是加号之间没有连接线
>>x = 0:pi/15:4*pi;
>>y = exp(2*cos(x));
>>plot(x,y,'r+')
>>
```

7. 设置默认的曲线形式

用户可以设置默认的曲线形式，这样，在绘制曲线时，可以不需定义线型而直接进行绘制。MATLAB 7.0 会将默认的曲线形式直接赋给所绘制的曲线。

例如 *set(0,'DefaultAxesLineStyleOrder',{'-o',':s','--+'})*命令将定义 3 种默认的曲线形式，要设置线型的颜色为灰白，可以使用 *set(0,'DefaultAxesColorOrder',[0.4,0.4,0.4])*。详细的信息用户可以参见 vcolorspec 的帮助信息。

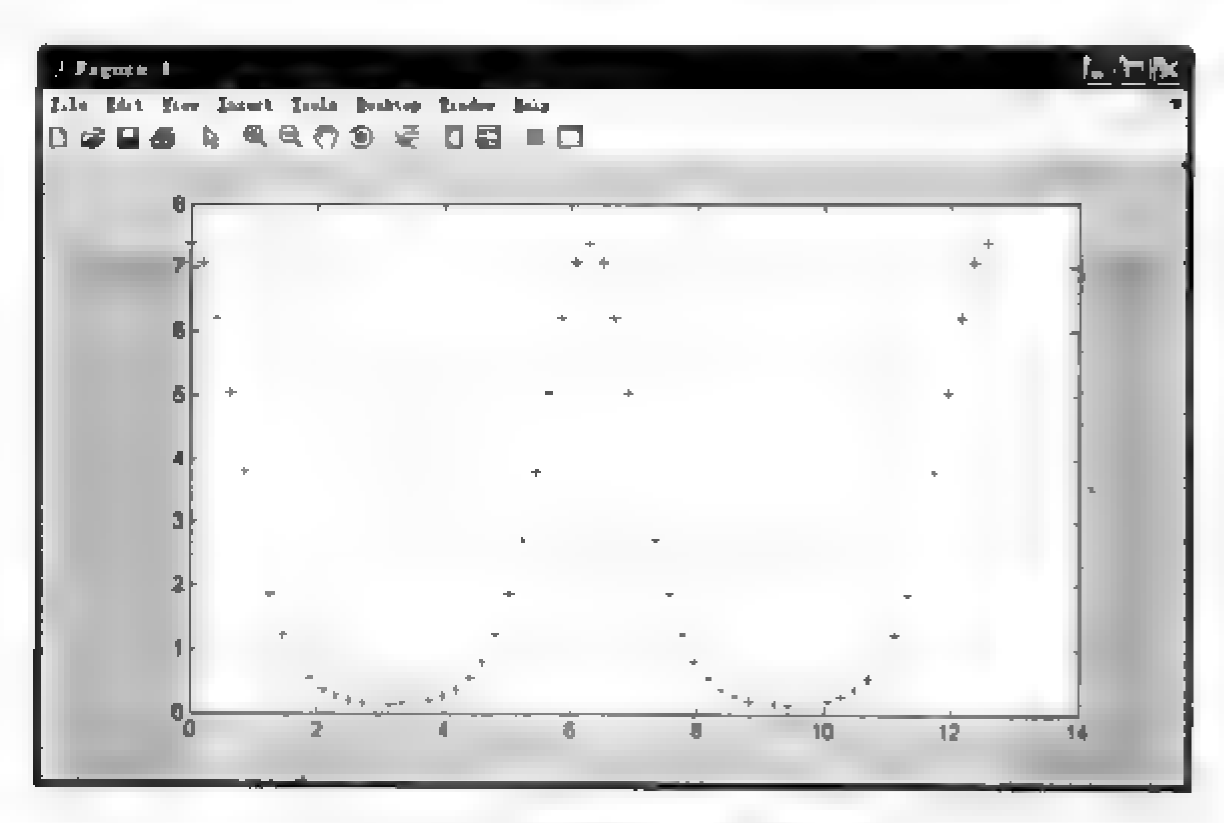


图 12-12　仅绘制数据点的二维图形

例 12-8　设置默认的曲线形式。

解：在命令窗口输入如下程序，并按 Enter 键确认输入，运行结果如图 12-13 所示。

```
>> set(0,'DefaultAxesLineStyleOrder',{'-o',':s','--+'})
>> set(0,'DefaultAxesColorOrder',[0.4,0.4,0.4])
>> x = 0:pi/10:2*pi;
>> y1 = sin(x);
>> y2 = sin(x-pi/2);
>> y3 = sin(x-pi);
>> plot(x,y1,x,y2,x,y3)
>>
```

所绘制的图形如图 12-13 所示。

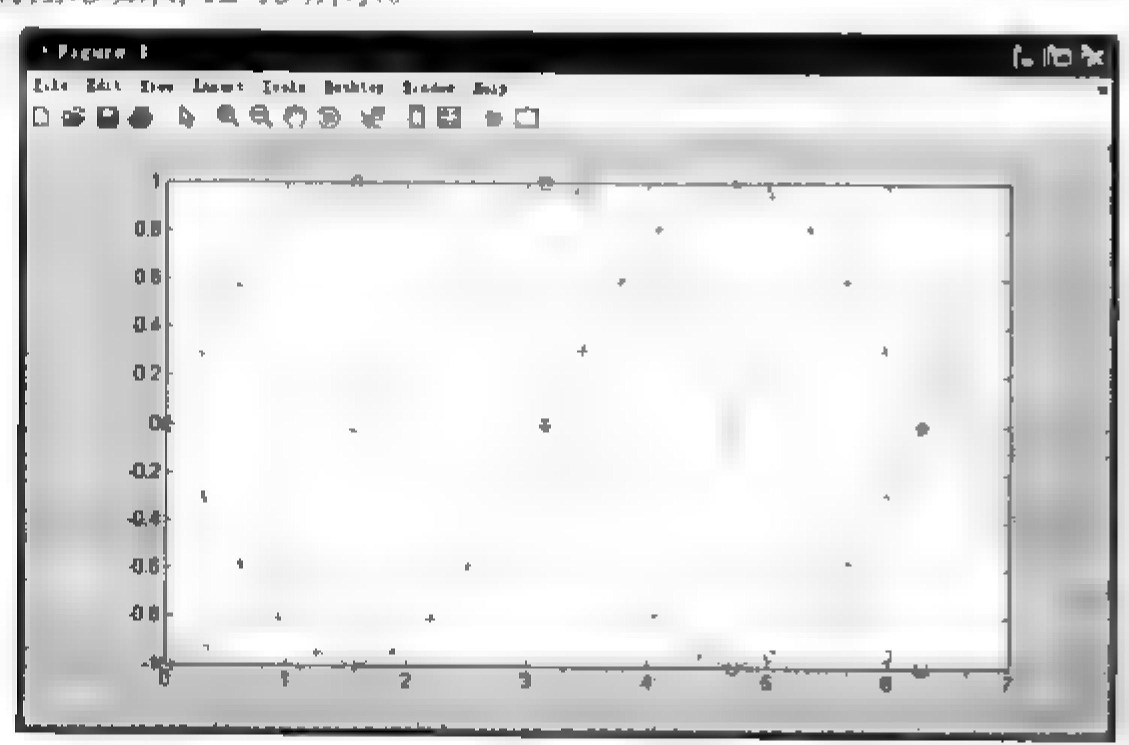


图 12-13　使用默认的曲线形式进行图形绘制

在用户结束 MATLAB 7.0 之前，默认的曲线形式将一直存在。如果用户要曲线默认的

曲线形式，可以使用如下的取消语句：

- *set(0,'DefaultAxesLineStyleOrder','remove')*
- *set(0,'DefaultAxesColorOrder','remove')*

8. 对数比例坐标轴和双Y轴

(1) 对数比例坐标轴

在进行数理统计时，使用传统的坐标系往往不能直观地看出统计模型的特征，此时，人们一般使用对数坐标系来绘制图形。MATLAB 7.0 语言提供了 loglog、semilogx 和 semilogy 等3个函数来进行这方面的图形绘制，在表12-1中，已经列出了它们的基本用法，这3个函数的用法和 plot 函数的用法完全相同，区别在于 loglog 函数对X轴和Y轴均采用对数坐标，而 semilogx 和 semilogy 函数则分别对X轴和Y轴采用对数坐标。

例12-9 使用 loglog、semilogx 和 semilogy 函数制函数的图形。

在命令窗口输入如下程序，并按 Enter 键确认输入，运行结果如图12-14所示。

```
>> % 该段程序介绍LOGLOG函数的用法
>> % 其中行x为自变量，y为因变量
>> %  x从1到100，y=exp(x)
>> x=linspace(1,100,100);
>> y=exp(x);
>> loglog(x,y)
>>
```

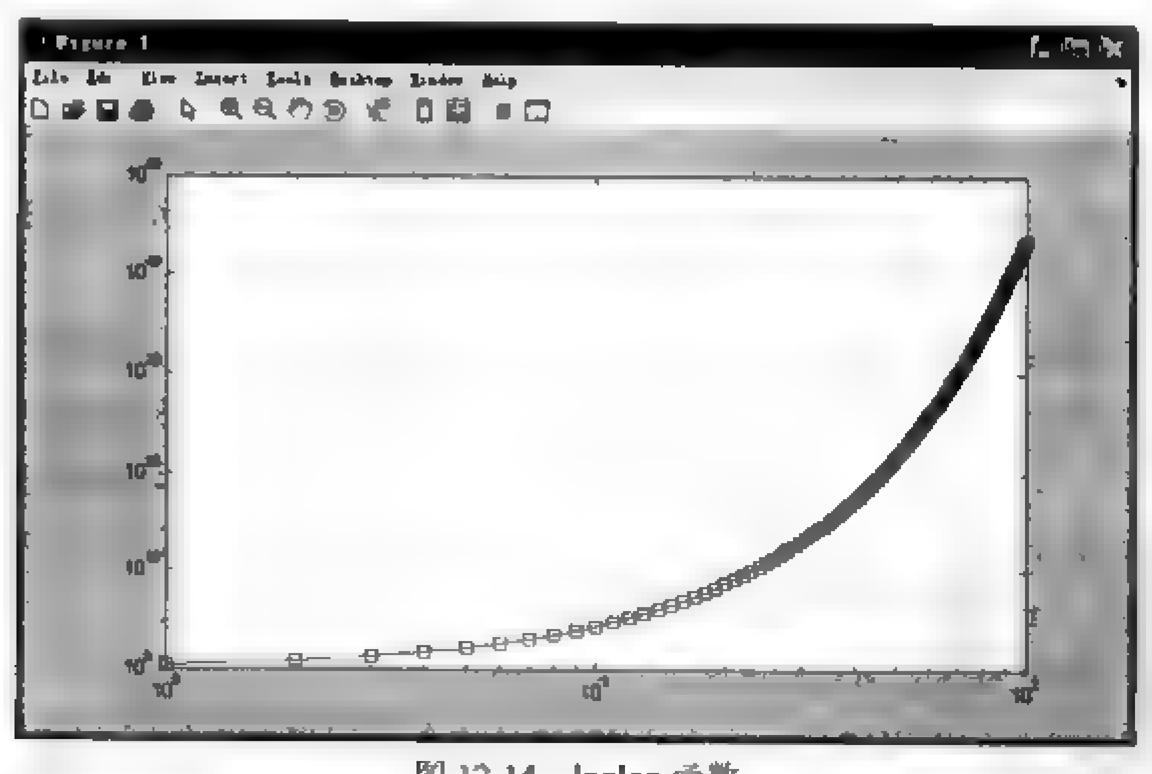


图12-14 loglog函数

继续在命令窗口输入如下程序，并按 Enter 键确认输入，运行结果如图12-15所示。

```
% 该程序用于绘制y坐标为对数坐标的图形
>> semilogy(x,y)
>>
```

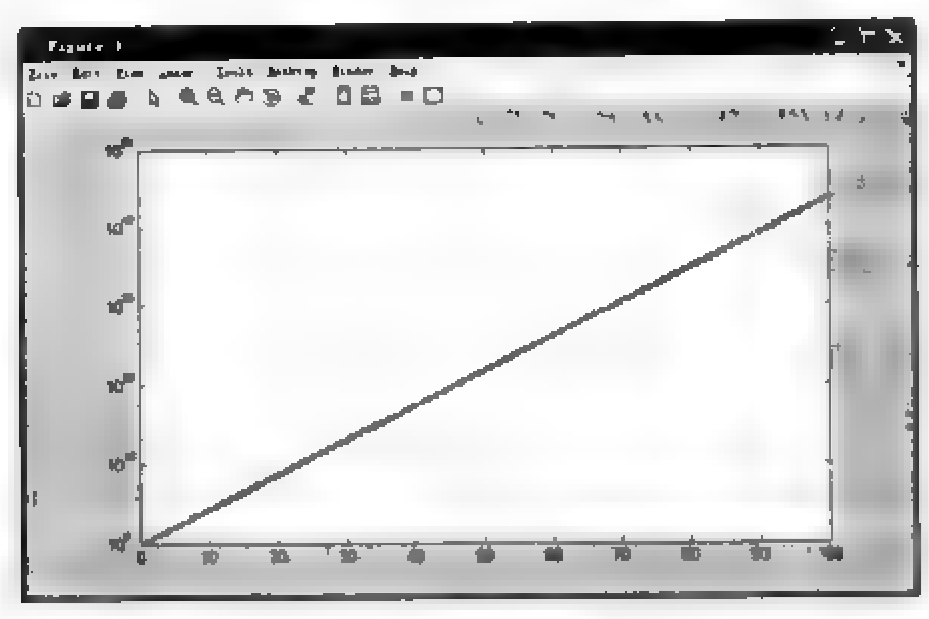


图 12-15　semilogy 函数

可见，由于 $y=exp(x)$，当 y 坐标使用对数坐标时，所绘图形成为一条直线。

同理可知 semilogx 函数的使用方法，在此不再赘述。

(2) 双 Y 轴的实现

在进行数值比较时，往往会遇到必须使用双纵坐标的情况，MATLAB 7.0 语言提供了 plotyy 函数来绘制双纵坐标二维图形。使用 plotyy 函数绘制的图形左右两端都显示有 Y 坐标轴。其使用格式如下。

- *plotyy*(*x*1, *y*1, *x*2, *y*2) 命令将 *x*1 和 *y*1 所对应的图形的纵坐标标注在图形的左边，并把 *x*2 和 *y*2 所对应图形的纵坐标标注在图形的右边。
- *plotyy*(*x*1, *y*1, *x*2, *y*2, *fun*) 命令可以选择绘制图形时所使用的形式，由 *fun* 函数来决定，*fun* 可以是@plot、@semilogx、@semilogy、@loglog 以及@stem 等。注意前边的@符号一定不能省略。
- *plotyy*(*x*1, *y*1, *x*2, *y*2, *fun*1, *fun*2) 命令使用 *fun*1(*x*1, *y*1) 来给左边的坐标轴绘制图形，并使用 *fun*1(*x*2, *y*2) 给右边的坐标轴绘制图形。
- 由于在使用 plotyy 函数图形的过程中，不能对所用曲线的属性进行设置，因此，用户要想对图中曲线的线型、颜色和数据点标识进行设置，就必须使用句柄图形控制来完成，具体内容将在后面的章节专门介绍。这里先介绍怎样调出图形控制句柄。
- [*ax*, *h*1, *h*2] = *plotyy*(...) 命令可以返回两条纵坐标的控制句柄，其中 *ax*(1) 是左轴的句柄，*ax*(2) 是右轴的句柄。

例 12-10　使用 plotyy 函数制函数双坐标图形。

在命令窗口输入如下程序，并按 Enter 键确认输入，运行结果如图 12-16 所示。

```
>> % 该段程序介绍 PLOTYY(X1,Y1,X2,Y2)命令的用法
>> % 其中 X1 为自变量，Y1= sin(X1)
>> %  X2 也为自变量，Y2= cos(X2)
>> x1=linspace(-7,7,100);
>> y1=sin(x1);
>> x2 linspace(-7,7,100);
>> y2=cos(x2);
```

```
>> plotyy(x1,y1,x2,y2)
>>
```

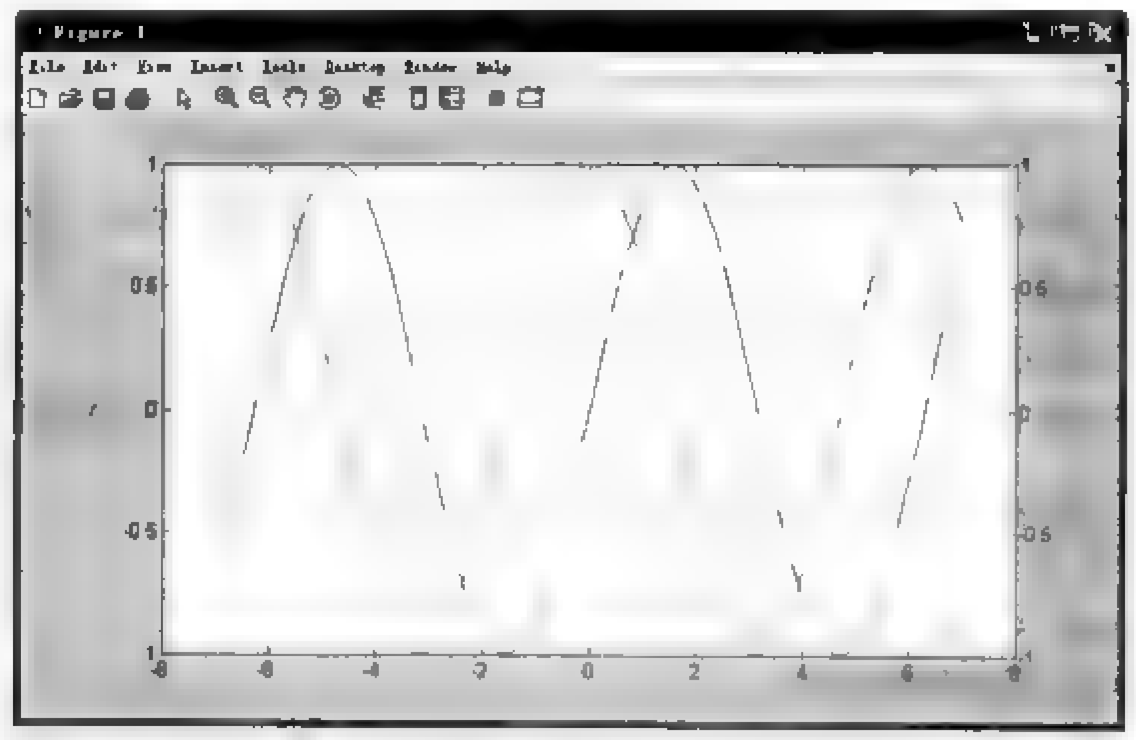


图 12-16 *plotyy*($x1, y1, x2, y2$) 命令

如果用户想使用不同的函数来绘制 X1-Y1 图形和 X2-Y2 图形，可以使用 *plotyy*($x1, y1, x2, y2, fun1, fun2$) 命令来完成绘制。

继续在命令窗口输入如下程序，并按 Enter 键确认输入，运行结果如图 12-17 所示。

```
>> % 该段程序介绍PLOTYY(X1,Y1,X2,Y2,FUN1,FUN2)命令的用法
>> % 其中X1为自变量，Y1=exp(X1)
>> %  X2也为自变量，Y2=0.01*(X2.^3+3*X2.^2+5*X2)
>> % 使用SEMILAGY函数绘制X1-Y1曲线，使用PLOT函数绘制X2-Y2曲线
>> x1=linspace(-2*pi,2*pi,100);
>> y1=exp(x1);
>> x2=linspace(-6,6,100);
>> y2=0.01*(x2.^3+3*x2.^2+5*x2);
>> plotyy(x1,y1,x2,y2,@semilogy,@plot)
>>
```

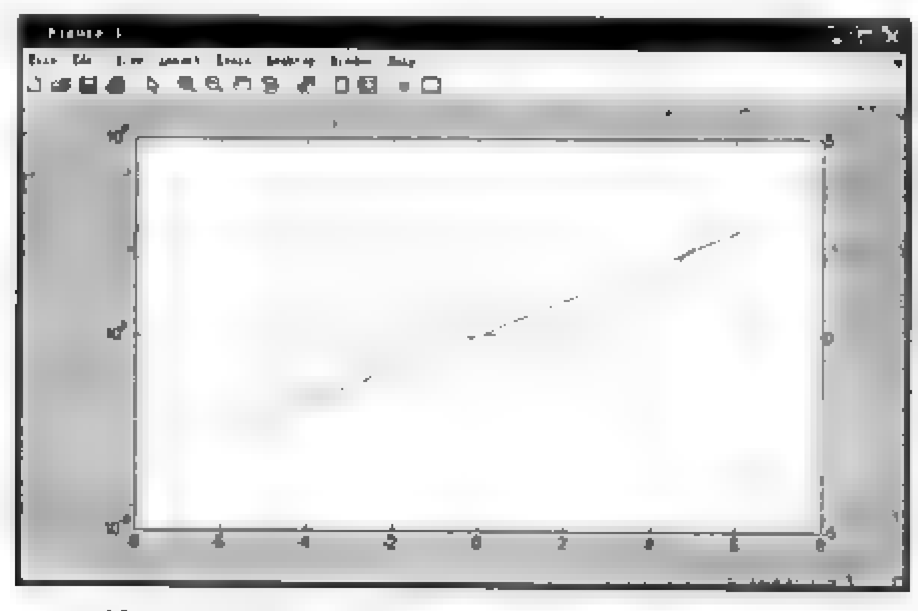


图 12-17 *plotyy*($x1, y1, x2, y2, fun1, fun2$) 命令

9. 极坐标图形的绘制

MATLAB 提供了 polar 函数来在极坐标下绘制图形，其一般的使用格式如下。

- *polar(theta,rho)* 命令使用极坐标，按照坐标的角度为 *theta*，极半径为 *rho* 绘制图形；
- *polar(theta,rho,s)* 命令同样使用极坐标，按照坐标的角度为 *theta*，极半径为 *rho* 绘制图形，但是在 S 中增加绘制图形的颜色和线型的定义。

例 12-11　使用 polar 函数在极坐标下绘制函数的图形。

解：在命令窗口输入如下程序，并按 Enter 键确认输入，运行结果如图 12-18 所示。

```
>> % 该程序介绍 polar 函数的用法
>> % 其中 t 为角度，s 为极半径
>> t=0:0.01:4*pi;
>> s=abs(sin(2*t).*cos(2*t));
>> polar(t,s,'-r+');
>>
```

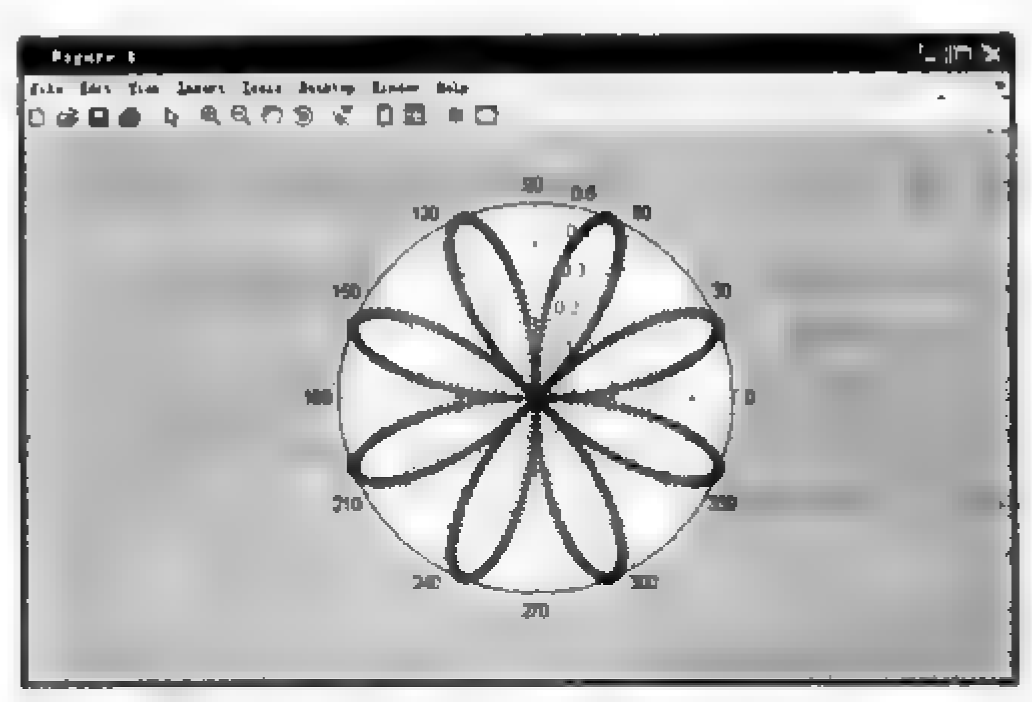


图 12-18　使用 polar 函数绘制极坐标图形

10. 多子图

用户在绘图过程中，还会经常需要将多个图形在同一窗口中显示出来，而不是简单地叠加。在 MATLAB 7.0 语言中，subplot 函数可以很好地实现这一功能。其通常的调用形式如下。

- $H = subplot(m,n,p)$ 或者 $subplot(mnp)$ 命令将图形窗口分解成 $m \times n$ 块绘图子域，并且设置第 p 块绘图子域作为当前绘图窗口；
- $subplot(m,n,p)$ 命令如果第 p 块绘图子域的轴系已经存在，则将其设置为当前绘图窗口；
- *subplot(m,n,p,'replace')* 命令如果第 p 块绘图子域轴系已经存在，则将其轴系删除，并且用 *'replace'* 代替；
- $subplot(m,n,P)$ 命令中，P 是一个向量时，将在所有的绘图子域中指定 P 中的

个坐标位置；

- *subplot('position',[left, bottomm, width, height])* 命令在标准的坐标系中，指定的位置处生成一个轴系。

例 12-12 使用 subplot 函数进行多图的绘制。

解：在命令窗口输入如下程序，并按 Enter 键确认输入，运行结果如图 12-19 所示。

```
>> x = 0:.1:20;
>> subplot(2,2,1)
>> plot(x,bessel(1,x),x,bessel(2,x),x,bessel(3,x));
>>
```

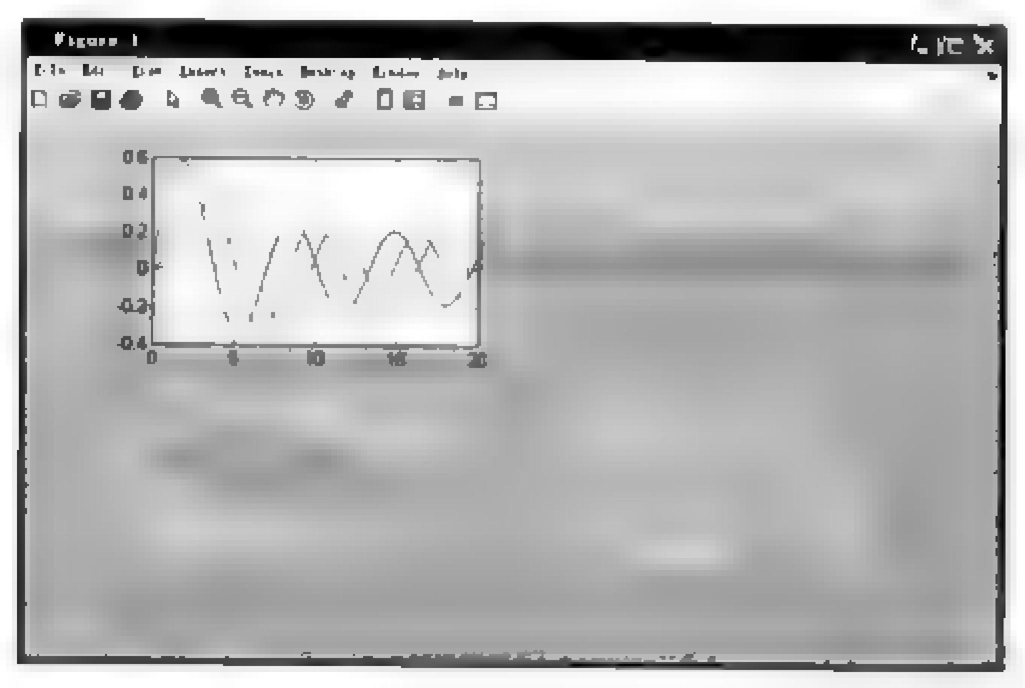


图 12-19 subplot(2,2,1)命令

继续在命令窗口输入如下程序，并按 Enter 键确认输入，运行结果如图 12-20 所示。

```
>> Z = peaks;
>> plot(Z)
>>
>>
```

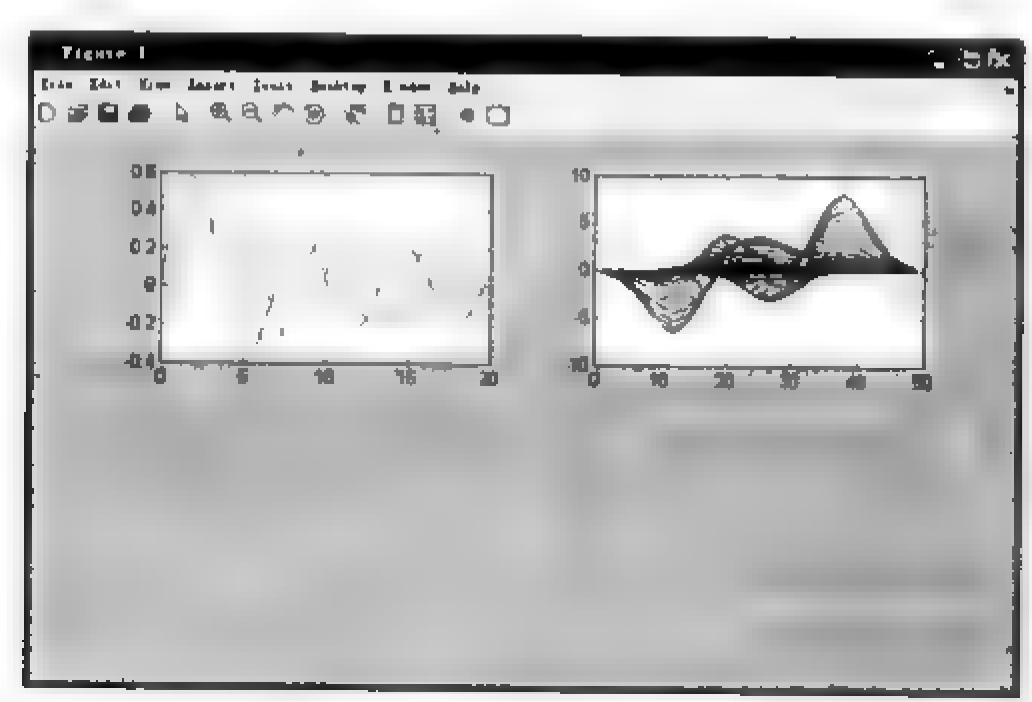


图 12-20 subplot(2,2,2)命令

继续在命令窗口输入如下程序，并按 Enter 键确认输入，运行结果如图 12-21 所示。

```
>> subplot(2,2,3)
>> y=cos(x);
>> S1='--b*';
>> plot(x,y,S1)
>> x=0:pi/15:4*pi;
>> y=sin(x);
>> S1='--b*';
>> plot(x,y,S1)
>> subplot(2,2,4)
>> x=3:0.01:5;
>> y=-x.^2+9*x-13;
>> plot(x,y)
>>
```

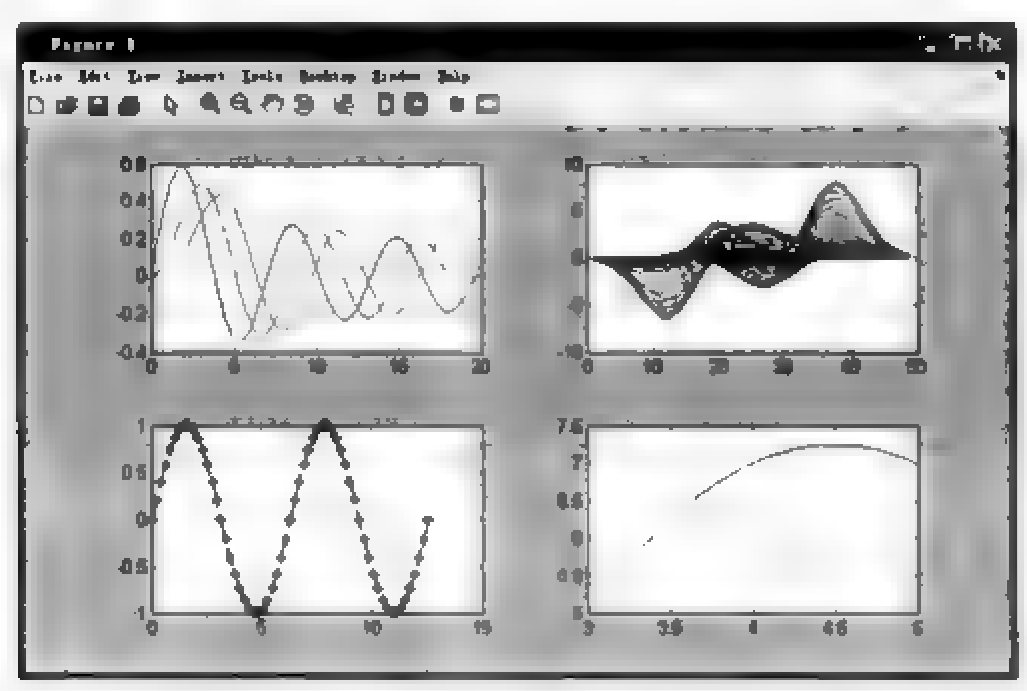


图 12-21　subplot(2,2,3)和 subplot(2,2,4)命令

在各个子图的绘制过程中，以下将要介绍的 xlabel、ylabel 和 title 等函数仍然有效，且对它们的使用都是针对其中的某个子图的，对其中的一个子图进行图形设置时不会影响到其他的子图。

12.1.3　图形注释

使用文本和其他说明性的材料对图形进行注释可以增加图形传递信息的能力。本节将介绍 MATLAB 7.0 中图形的注释功能，主要包括如下几个部分的内容：

- 在图形的任意地方添加文本、直线、箭头、长方形、椭圆以及其他的注释方式；
- 将注释定位于数据空间；
- 增加文本注释和颜色条；
- 增加坐标轴标签和图形标题；
- 对图形体的属性进行编辑。

可以使用4种方式对图形进行注释，即分别通过调用图形注释工具栏中的图标、图形调色板中的注释工具、Insert 菜单中的注释命令和直接使用注释命令来对图形进行注释。下面对这4种方式分别予以介绍。

(1) 图形注释工具栏的显示

选择 View 菜单下的 Plot Edit Toolbar 命令，如图12-22所示，调出如图12-23所示的图形注释工具栏。

图12-22 调出图形注释工具栏

工具栏中各个图标的功能如图12-23所示。

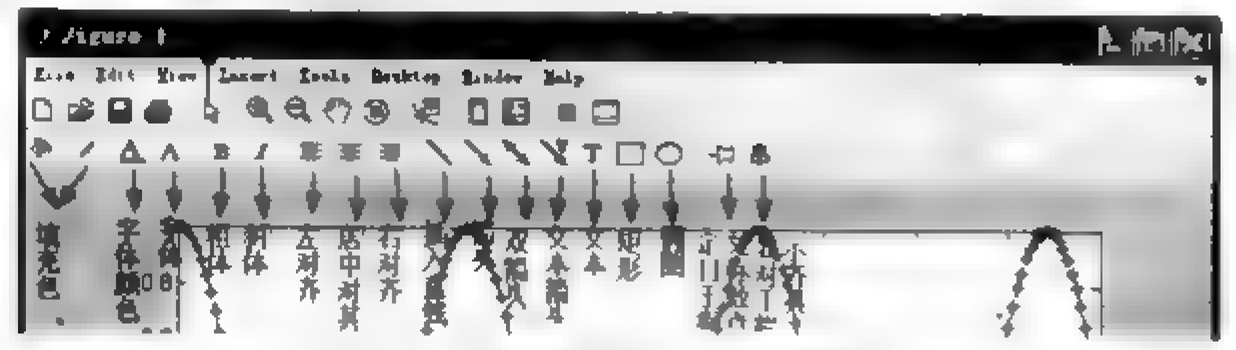


图12-23 图形注释工具栏的功能简介

(2) 图形调色板中的注释工具

基本的注释工具也可以从图形调色板中调出，可以选择 View 菜单下的 Figure Palette 命令可以调出图形调色板，如图12-24所示，图中红线框起的部分即为注释工具。

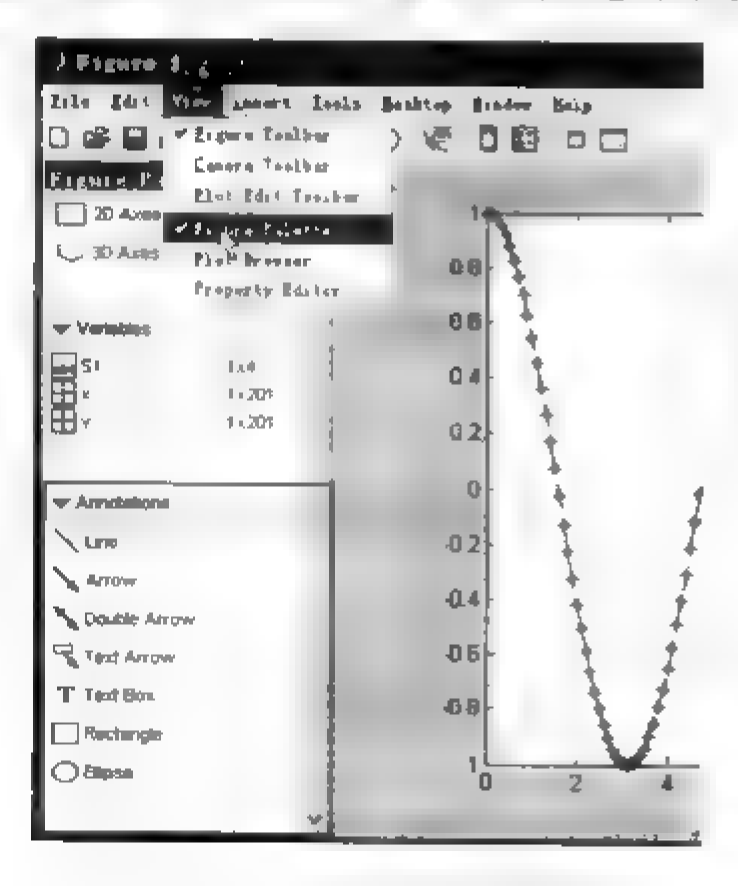


图12-24 图形调色板中的注释工具

(3) 从 Insert 菜单增添注释

用户也可以从 Insert 菜单增添注释，打开 Insert 菜单，从 Insert 下拉菜单中选择用户需要的注释种类即可，如图 12-25 所示。

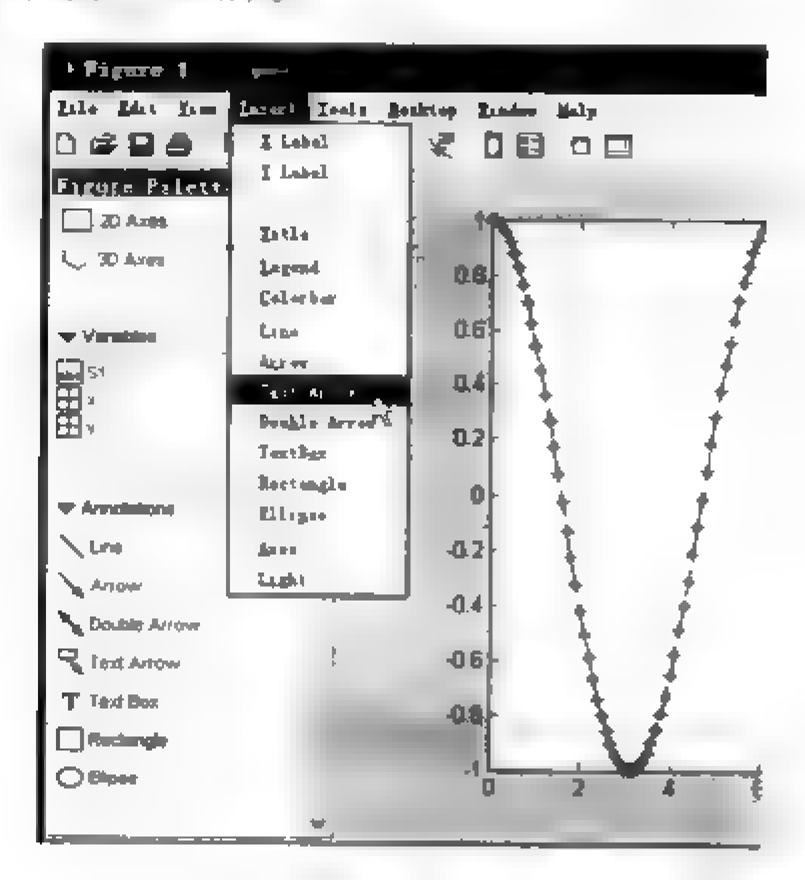


图 12-25　从 Insert 菜单增添注释

(4) 使用命令语句增添注释

用户也可以直接通过调用命令语句来增添注释语句，表 12-4 列出了 MATLAB 7.0 中的注释命令。

表 12-4　MATLAB 7.0 的注释命令语句

函　数	功能描述
annotation	创建线、箭头、文本箭头、双箭头、文本框、矩形和椭圆
xlabel, ylabel, zlabel	给相应的坐标轴增添标签
title	给图形增添标题
colorbar	给图形增添颜色条
legend	给图形增添图例

下面对各种图形的注释方式进行介绍，用户在学完本节内容之后，将对图形注释具有比较深刻的了解，从而能够完成比较复杂的图形注释。

1. 图题的标注

在 MATLAB 7.0 中，图题是位于坐标轴顶部的一个文本字符串，一般来说，图题定义的是一个图形的主题。

在 MATLAB 中，通常可以使用 3 种方式给图形添加图题。

- 使用 Insert 菜单中的 Title 命令；

◆ 使用属性编辑器(Property Editor)；
◆ 使用 title 函数。

下面对这3种添加图题的方式予以简单介绍。

(1) 使用 Insert 菜单中的 Title 命令添加图题

① 打开 Insert 菜单并选择 Title 命令，MATLAB 7.0 将在图框的顶部打开一个文本框。要注意的是，当用户选择了 Title 命令时，MATLAB 7.0 将自动切换到图形编辑模式。

② 在文本框里边输入该图的图题

③ 输入完图题后，在图形背景的任意地方单击即可关闭该文本框。如果用户单击图形窗口中的其他对象，例如轴或是线，那么在关闭该文本框的同时，将自动切换到用户所选择的对象。

④ 欲改变图题的字体，用户可以像编辑图形中的其他字体一样对图题中的字体进行编辑。

(2) 使用属性编辑器(Property Editor)添加图题

① 打开 Tools 菜单，选择 Edit Plot 命令，激活图形编辑状态。

② 在图形框内双击空白区域，调出属性编辑器，如图 12-26 所示；也可以采取在图形框内右击，从弹出的快捷菜单中选择 Properties 项的方式调出属性编辑器；或者是在 View 菜单中选择 Property Editor 项。

图 12-26 在属性编辑器(Property Editor)中添加图题

③ 在 Title 输入框中添加图题

用户可以修改图题的字体、位置和其他的属性。

◆ 要移动图题，选择图题文本框并拖动它到相应位置。
◆ 要编辑图题文字，双击图题并输入新的字符。
◆ 要改变字体或其他文本属性，可以选择图题并右击来显示 context 菜单。

(3) 使用 title 函数添加图题

用户也可以使用 title 函数给图形添加图题，其使用格式如下。

◆ *title*('*text*') 命令将 *text* 增加到当前图形的顶边部分；
◆ *title*('*text*','*property*1' *propertyvalue*1,'*property*2'...) 命令给标题的各个属性赋值。

例 12-13 给 MATLAB 7.0 的图形添加标题。

解：在命令窗口中输入如下语句，并按 Enter 键确认。

```
>> x=linspace(-2*pi,2*pi,100);
```

```
>> y=sin(x);
>> plot(x,y)
>>
```

此时，生成了一条正弦曲线，如图 12-27 所示。

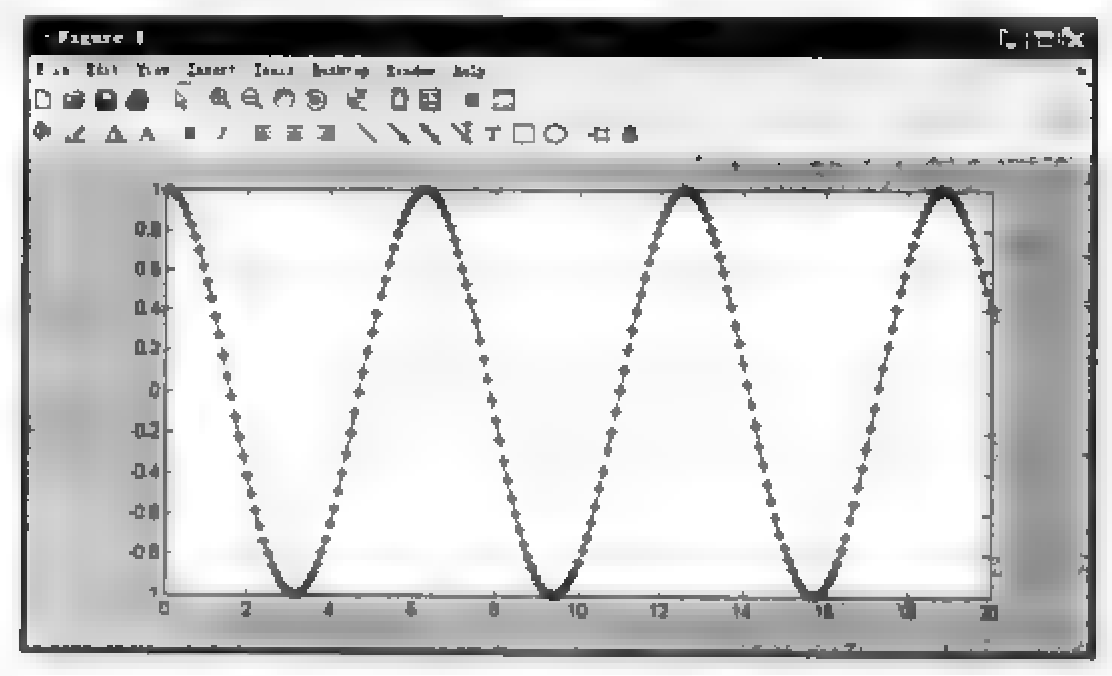


图 12-27　生成的正弦曲线

下面再为这条正弦曲线添加图题，使用上文提到的第 2 种方法，即首先打开 Tools 菜单，单击 Edit Plot 命令，激活图形编辑状态。然后在图形框内双击空白区域，调出属性编辑器，在 Title 输入框中添加图题“正弦曲线”单击属性编辑器右边的 Inspector…按钮，图题即被添加到图形上，如图 12-28 所示。

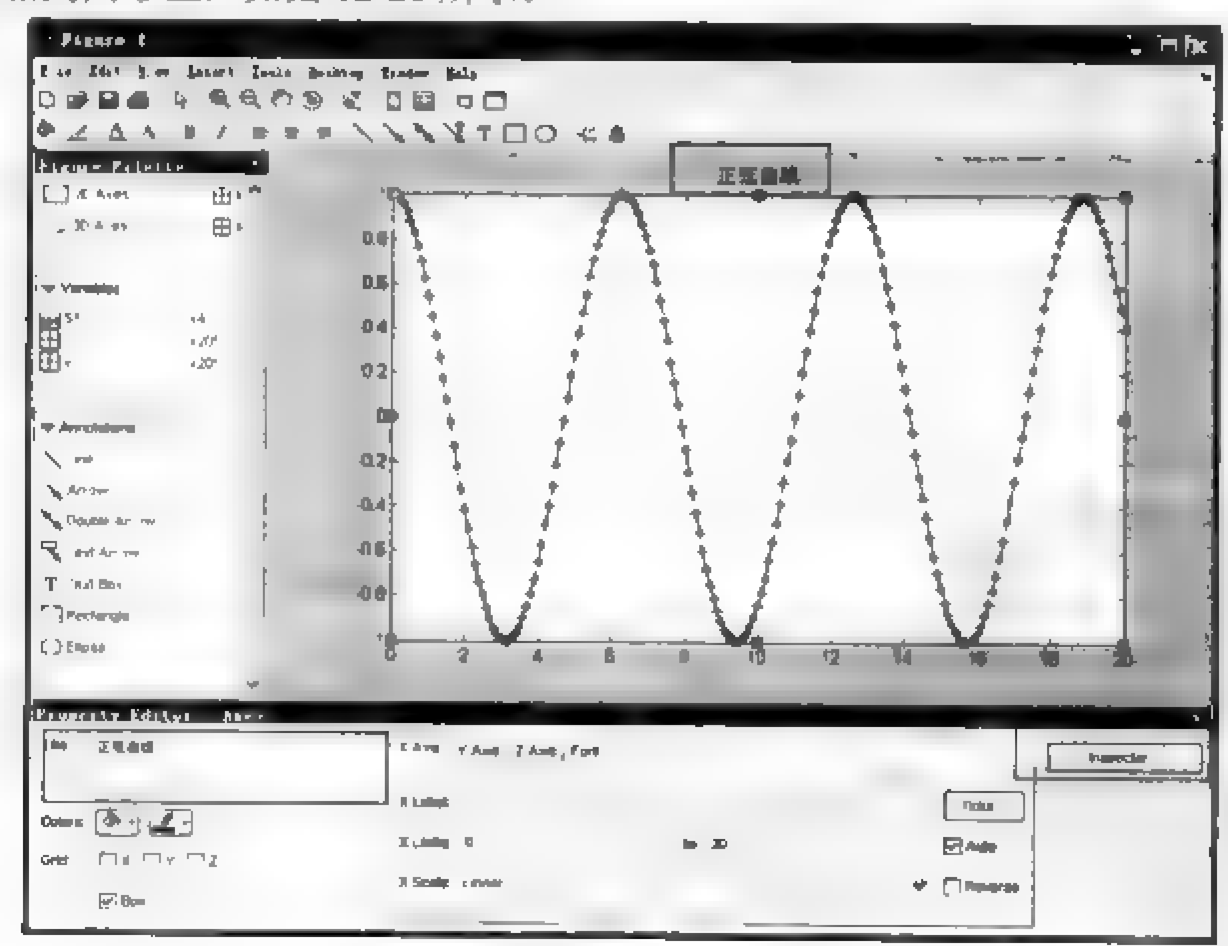


图 12-28　图题的添加

用户也可以使用其他两种方式给图形添加图题，详细的使用方法可以参见 MATLAB 的帮助系统，这里不再赘述。

2. 坐标轴的标签

前边所绘制的图形虽然也有坐标轴，坐标轴上也标有数字。但是，与用户通常所绘的图形相比，这样的标注过于简单，MATLAB 7.0 也可以给坐标轴设置标签，这些标签是与 x、y 或是 z 轴对齐的字符串，坐标轴的标签主要用于解释各坐标轴的单位信息。

在 MATLAB 7.0 中，可以使用如下 3 种方式给图形的坐标轴添加标签。

- 使用 Insert 菜单下的 Label 选项；
- 使用属性编辑器(Property Editor)；
- 使用 MATLAB 7.0 的添加标签命令；

下面对这 3 种添加坐标轴标签的方法予以简单介绍

(1) 使用 Insert 菜单下的 Label 选项添加坐标轴的标签

操作步骤如下。

① 打开 Insert 菜单，选择该菜单栏下的标签选项：X Label、Y Label 或是 Z Label。MATLAB 7.0 将沿着对应的坐标轴打开一个文本框，如图 12-29 所示。由于本图是二维图形，所以 Z Label 为灰色，表示该命令处于不可选状态。

图 12-29 使用 Insert 菜单下的 Label 选项添加坐标轴的标签

② 输入标签的内容，或是对已有的标签进行编辑。当对 y 轴和 z 轴的标签进行编辑的时候，在输入状态下这些标签都处于水平状态，输入完成之后 MATLAB 7.0 将自动把它们调整为与坐标轴对齐的状态。

③ 在输入完标签的内容后，在图形背景的任意地方单击即可关闭该文本框。如果用户单击图形窗口中的其他对象，如图题或线，那么在关闭该文本输入框的同时，将自动切换到用户所选择的对象。

(2) 使用属性编辑器(Property Editor)添加坐标轴标签

操作步骤如下。

① 打开 Tools 菜单，选择 Edit Plot 命令，激活图形编辑状态。

② 在图形框内双击空白区域，调出属性编辑器，如图 12-30 所示；也可以采取在图形框内右击，从弹出的菜单中选择 Properties 项的方式调出属性编辑器；或者是在 View 菜单中选择 Property Editor 项。

图 12-30 在属性编辑器(Property Editor)中添加坐标轴标签

③ 根据用户所需要添加标签的轴，选择 X Axis、Y Axis 或是 Z Axis 模块，并添加文本。

此外，用户还可以使用属性编辑器来对坐标轴的标签进行旋转，具操作步骤如下。

① 打开 Tools 菜单，选择 Edit Plot 命令，激活图形编辑状态。

② 单击需要选择的坐标轴标签，调出属性编辑器，并右击所选的文本，从弹出的属性编辑器中单击 Inspector…按钮，如图 12-31 所示。

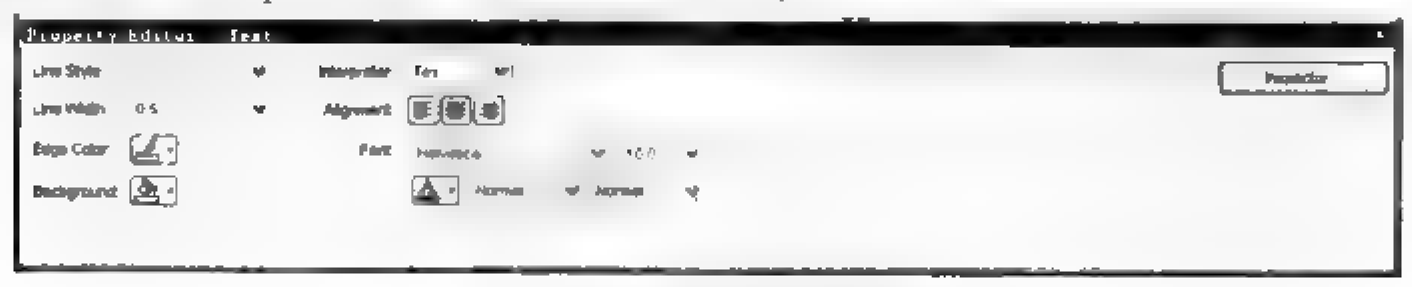


图 12-31　坐标轴标签的属性编辑器

③ 单击 Inspector…按钮调出 Property Inspector，如图 12-32 所示。

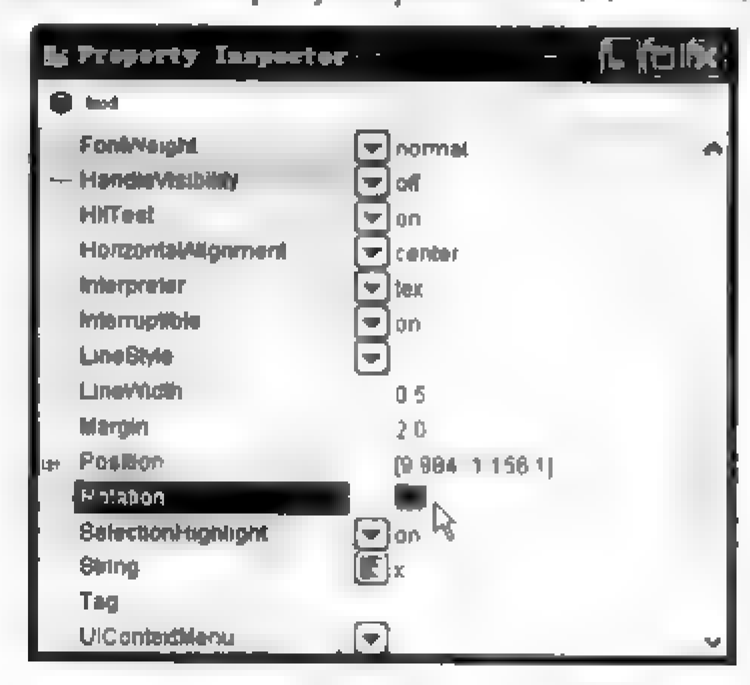


图 12-32　Property Inspector 对话框

④ 在 Property Inspector 中选择 Rotation 选项，默认的 Rotation 值为 0，代表坐标轴的标签与坐标轴对齐。用户可以修改 Rotation 的数据，这样可以相应地改变标签的角度。

(3) 使用坐标轴标签命令添加坐标轴标签

用户可以使用 xlabel、ylabel 和 zlabel 命令分别给 x 轴、y 轴和 z 轴添加标签。具使用命令如下。

- *xlabel*('*text*') 命令将 text 增加到当前 x 轴的旁边；
- *xlabel*('*text*','*property*1' *propertyvalue*1,'*property*2'...) 命令给 x 轴标注的各个属性赋值。
- *ylabel*('*text*') 命令将 text 增加到当前 y 轴的旁边；
- *ylabel*('*text*','*property*1' *propertyvalue*1,'*property*2'...) 命令给 y 轴标注的各个属性赋值。
- *zlabel*('*text*') 命令将 text 增加到当前 z 轴的旁边；
- *zlabel*('*text*','*property*1' *propertyvalue*1,'*property*2'...) 命令给 z 轴标注的各个

属性赋值。

例 12-14 使用 xlabel 和 ylabel 命令给已知图形添加坐标轴标注。

解：在命令窗口中输入如下语句，并按 Enter 键确认。

```
>> x=linspace(1,100,100);
>> y=log(x);
>> plot(x,y)
>> title('自然对数图');
>> xlabel('x 取值范围是 1—100');
>> ylabel('y=log(x)');
>>
```

生成的图形如图 12-33 所示。

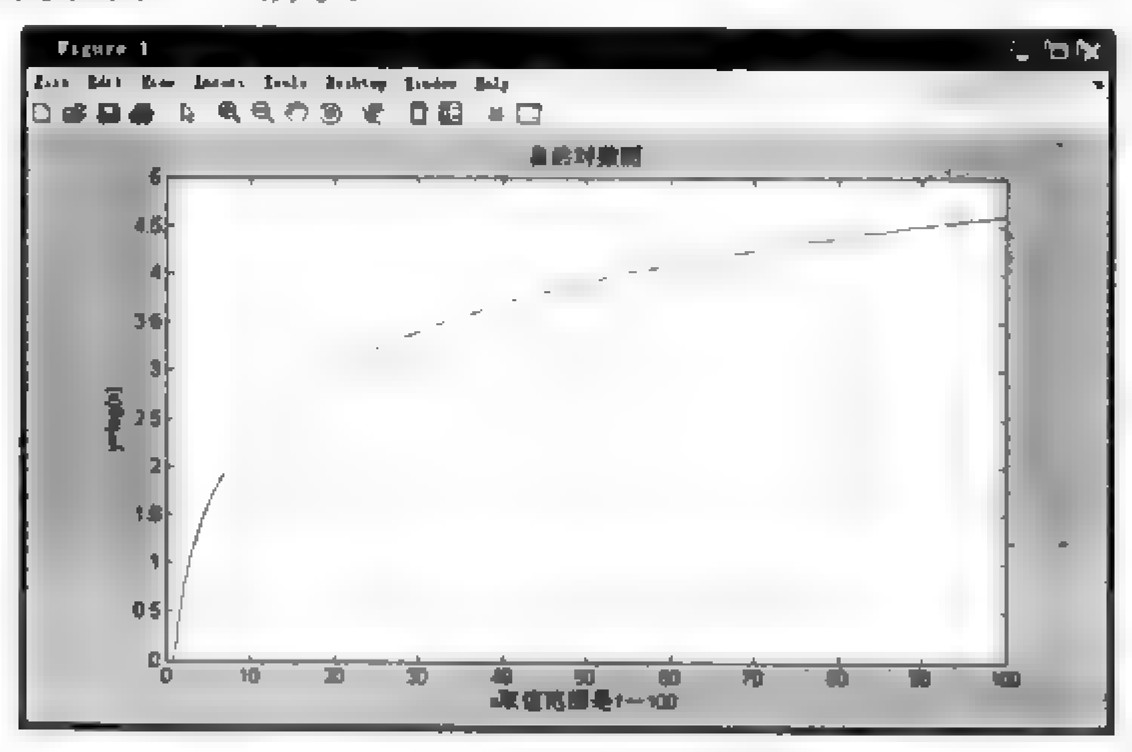


图 12-33 xlabel 函数和 ylabel 函数的使用

3. 文本标注和交互式文本标注

用户可以在 MATLAB 7.0 图形窗口的任意地方添加文本注释，从而更好地解释图形窗口的数据。MATLAB 7.0 提供了 text 函数和 gtext 函数来进行文本标注。其中 gtext 函数的使用形式更为灵活，可以实现交互式文本标注。下面对它们分别予以介绍。

(1) 使用 text 函数进行文本标注

MATLAB 7.0 语言提供了函数 text 图形进行文本标注，它的通常调用形式如下：

- *text*(*x*, *y*, '*string*') 在坐标为(*x*, *y*)的位置上添加文本标注'*string*'。如果 *x* 和 *y* 是向量，那么 text 函数将在所有这些位置点上都进行文本标注。如果字符串'*string*'和 *x*、*y* 具有相同的长度，text 函数将把字符串'*string*'的字符标注到 *x* 和 *y* 的相应位置上；
- *text*(*x*, *y*, *z*, '*string*') 命令将文本标注添加到三维坐标中。

例 12-15 使用 text 函数给已知图形添加文本标注。

解：在命令窗口中输入如下语句，并按 Enter 键确认。

```
>> x=linspace(-5,5,100);
>> y=x.^4-22*x.^2-6*x+10;
>> plot(x,y)
>> title('多项式图形')
>> xlabel('x 的取值范围')
>> ylabel('y 的值')
>> text(0,10,'y=x.^4-22*x.^2-6*x+10')
>>
```

生成的图形如图 12-34 所示。

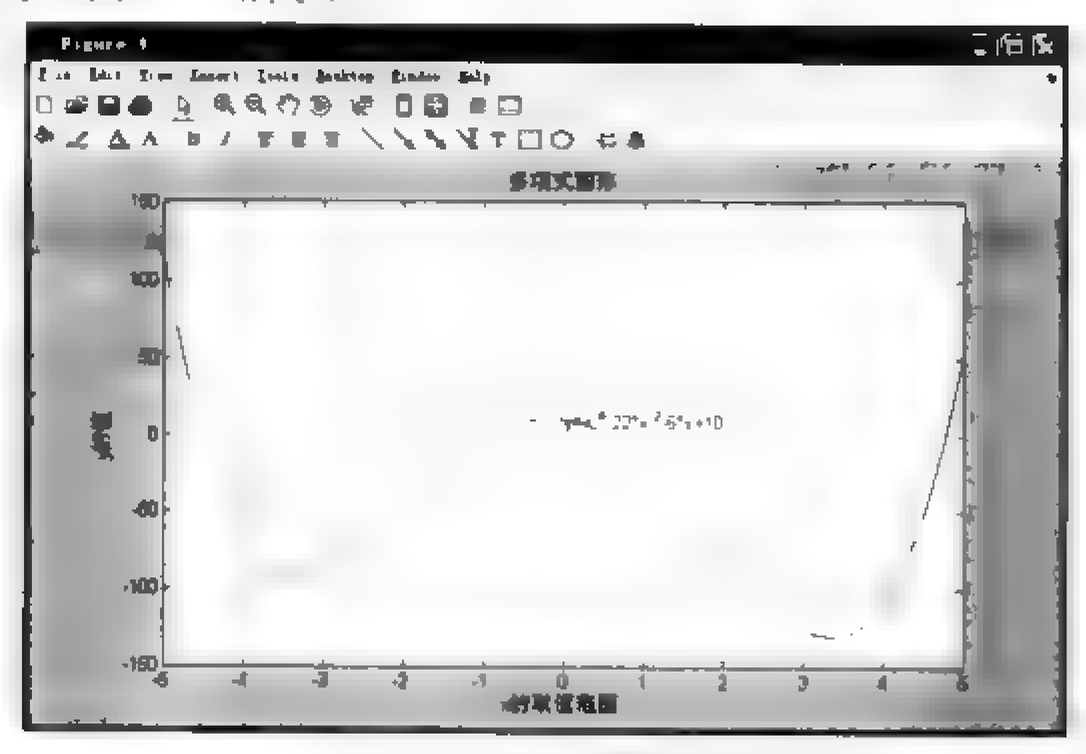


图 12-34　使用 text 函数添加文本标注

(2) 使用 gtext 函数进行交互式文本标注

MATLAB 7.0 语言还可以以图形进行交互式文本标注，所提供的函数为 gtext，它的通常调用形式如下。

- *gtext('string')* 命令将在图形窗口中显示一个"十"字交叉线，用户可以通过移动鼠标或使用键盘来选择文本标注的位置，当选定位置后，单击鼠标，系统将把指定的文本显示到所选择的位置上；
- *gtext(c)* 命令将单元型矩阵的各个子单元中的字符显示到指定的位置上；
- *gtext(...,'propertyname',propertyvalue,...)* 命令给指定的文本属性赋值，可以使用单个说明语句给多行标识赋值。

例 12-16　使用 gtext 函数给已知图形添加文本标注。

解：在命令窗口中输入如下语句，并按 Enter 键确认。

```
>> % 本程序显示 GTEXT 函数的用法
>> % 用户可以先使用 plot 函数绘出图形，再使用 gtext 函数进行文本标注
>> x=0:0.1:5*pi;
>> y=cos(x);
>> plot(x,y)
```

```
>> gtext('y=cos(x)','fontsize',14,'fontweight','bold','fontname','楷体')
>>
```

生成的图形如图 12-35 和 12-36 所示。

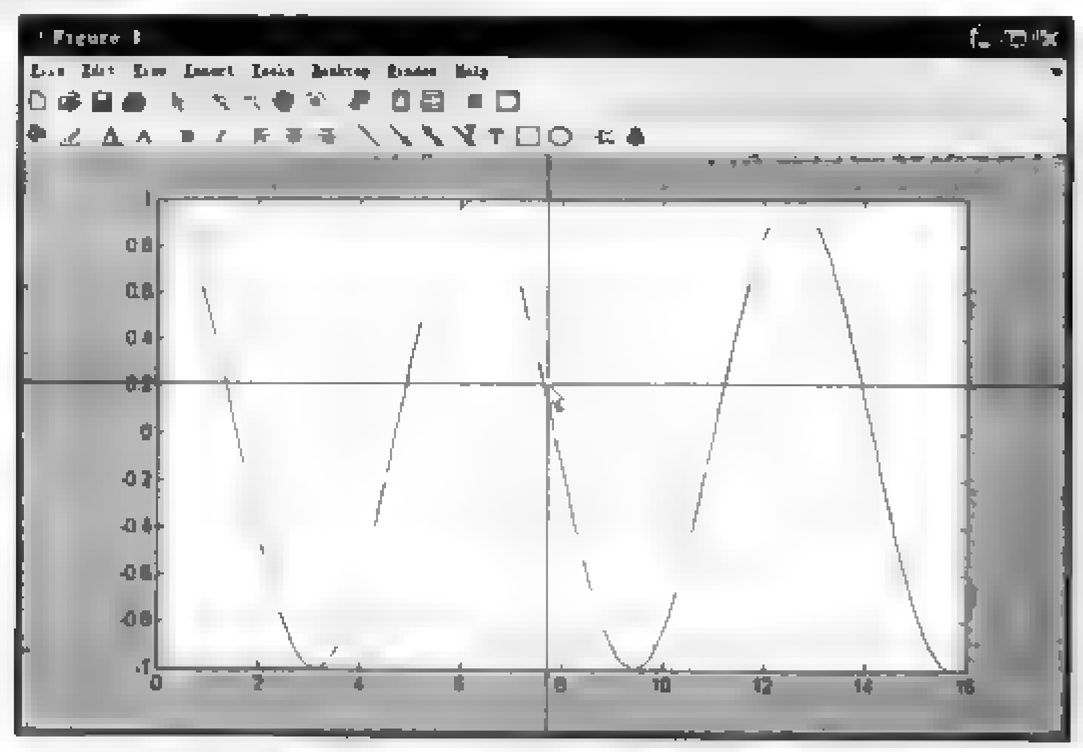


图 12-35 等待确认的“十”字形

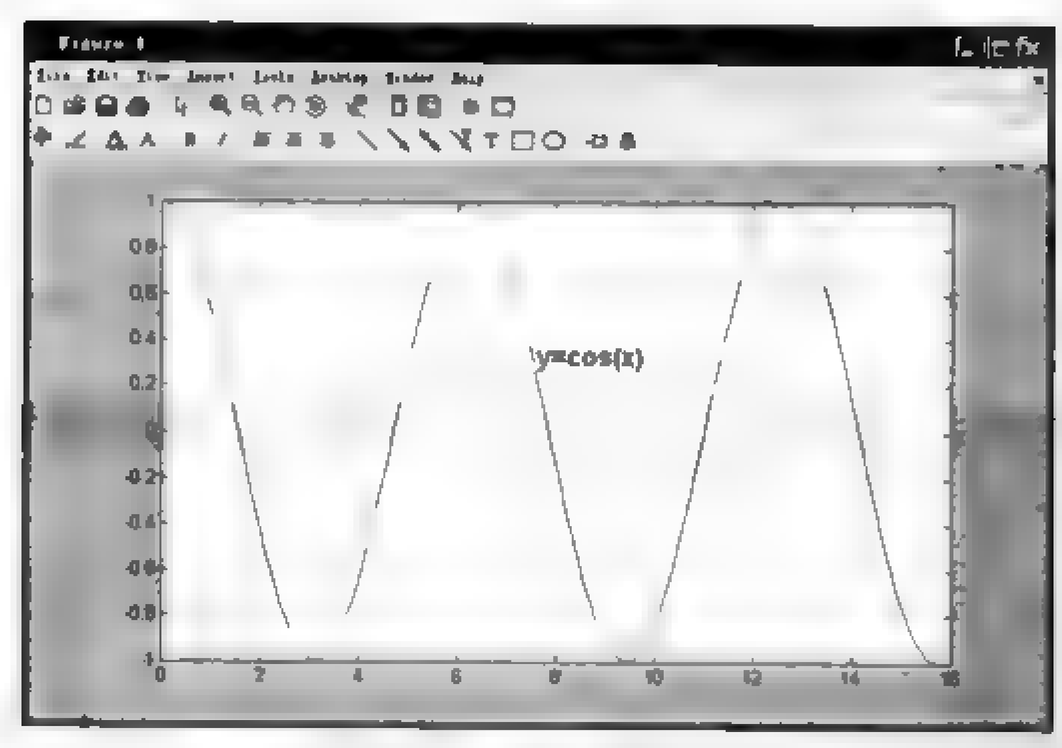


图 12-36 gtext 函数的调用结果

4. 在图形编辑模式下添加箭头和直线

(1) 箭头和直线的生成

在图形编辑模式下，用户可以在图形窗口的任意位置添加箭头和直线，由于添加箭头和直线的操作步骤基本一致，所以在这里一起介绍。

用户也可以使用 text 命令生成箭头符号，方法是在 text 命令的文本框中输入箭头符号。但是，这种方法生成的箭头只能指向水平方向的左边或是右边，并不能任意设定方向。通常使用的添加箭头和直线的方式是从 Insert 菜单调用或是直接从图形注释工具栏调用。下面介绍一下添加箭头和直线的步骤。

① 打开 Insert 菜单并选择 Arrow 或 Line 选项，或是在图形注释工具栏里单击 Arrow 或 Line 按钮。MATLAB 7.0 会将光标转换为一个十字线。

② 将光标定位于图形中箭头或是直线要开始的位置，按住鼠标的左键或右键不放，移动鼠标来定义箭头或直线的长度和方向。

③ 释放鼠标左键或右键完成添加操作。

MATLAB 7.0 语言还可以使用 line 函数在图形中添加新的标注直线，它的通常调用形式如下。

line(x, y) 命令在当前图形窗口中添加以向量 x 和向量 y 绘制而成的直线。

例 12-17　使用 line 函数给已知图形添加直线标注，并使用 Insert 菜单给图形添加箭头标注。

解：在命令窗口中输入如下语句，并按 Enter 键确认。

```
>> % 本程序显示 LINE 函数的用法
>> % 用户可以先使用 plot 函数绘出图形，再使用 LINE 函数进行添加直线标注
>> x=0:0.2:5*pi;
>> y=sin(x);
>> plot(x,y)
>> X=[0 16];
>> Y=[0 0];
>> line(X,Y)
>>
```

此时，直线已经被添加到图形中，如图 12-37 所示，下面再使用 Insert 菜单给图形添加箭头标注。

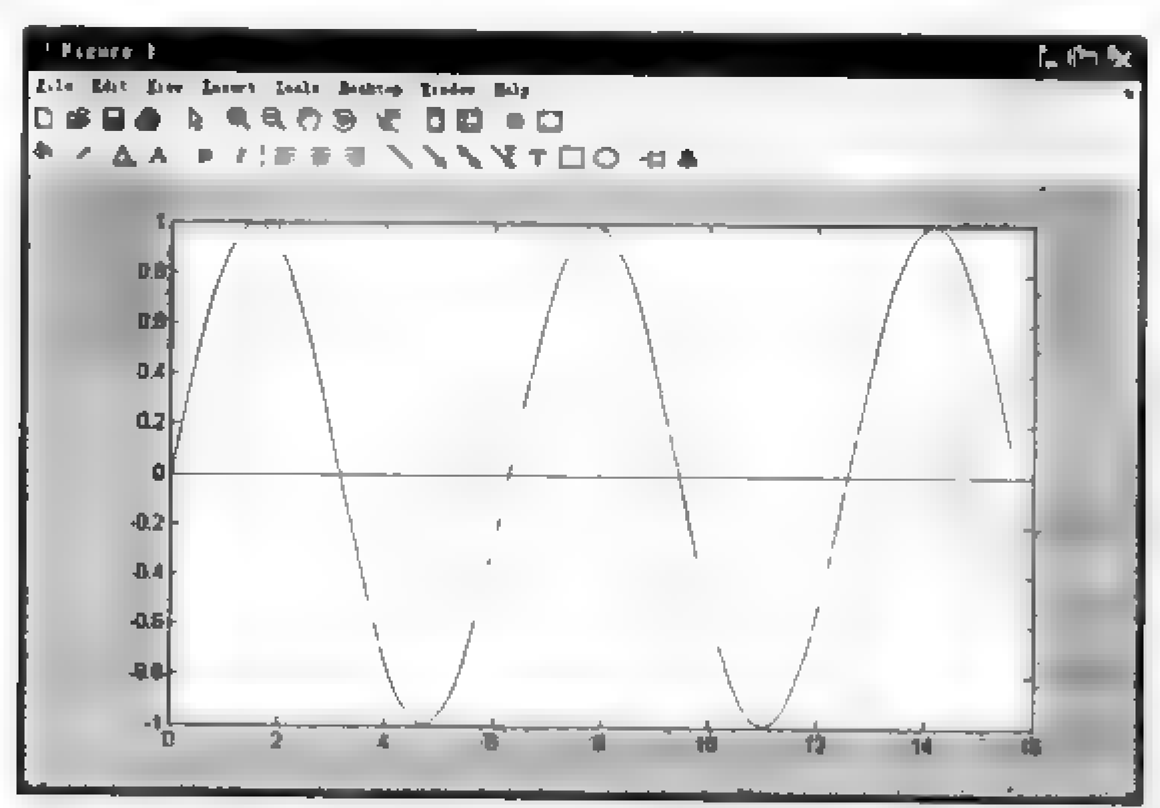


图 12-37　使用 line 函数给图形添加直线

按照上文所示的步骤，在图形窗口的任意地方可以添加箭头，如图 12-38 所示。

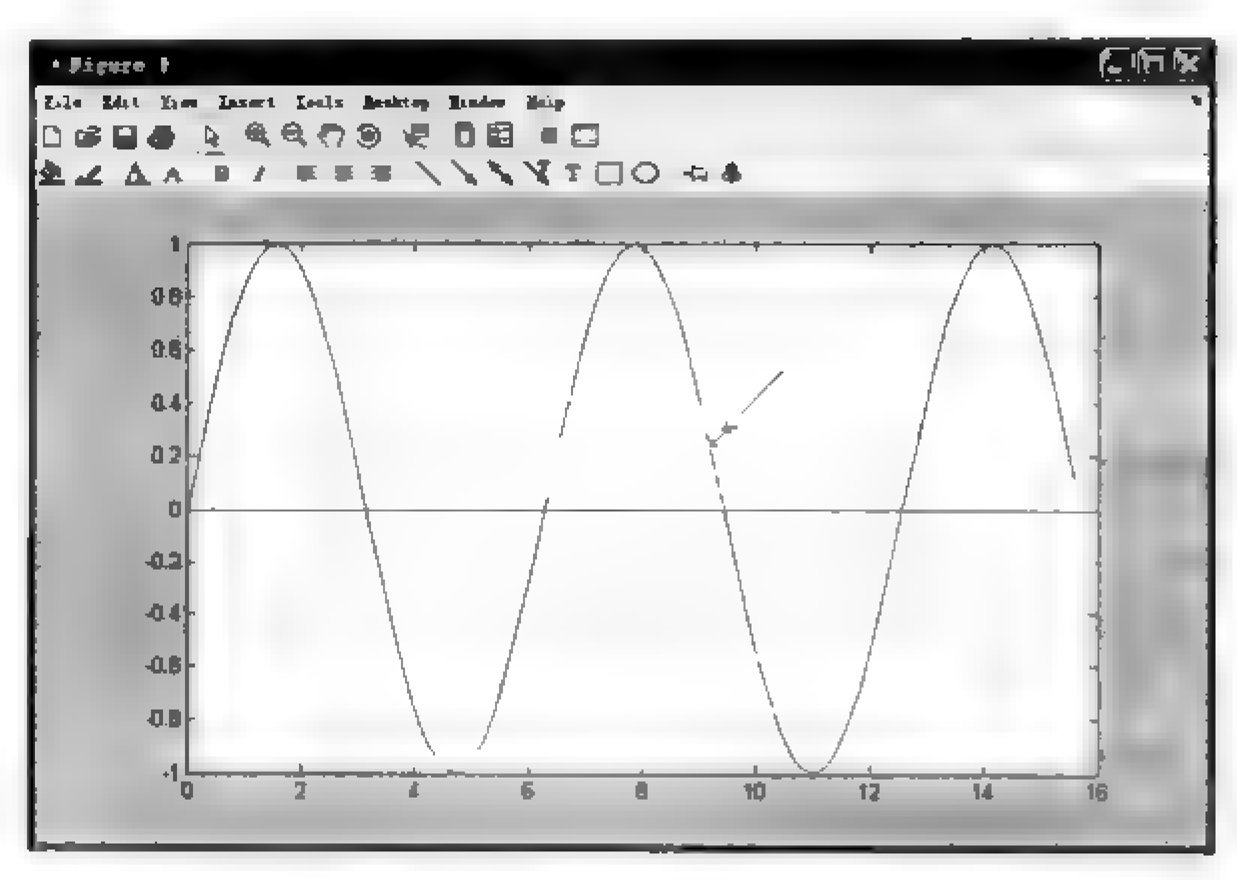


图 12-38　在图形窗口中添加箭头

添加完箭头之后，用户可以继续进行其他操作，比如在箭头的尾部再添加文本框等。

(2) 箭头和直线的编辑

用户可以使用编辑菜单来对箭头和直线的形状进行编辑。当处于图形编辑状态下时，右击箭头或直线可以调出它们的编辑菜单，如图 12-39 所示。

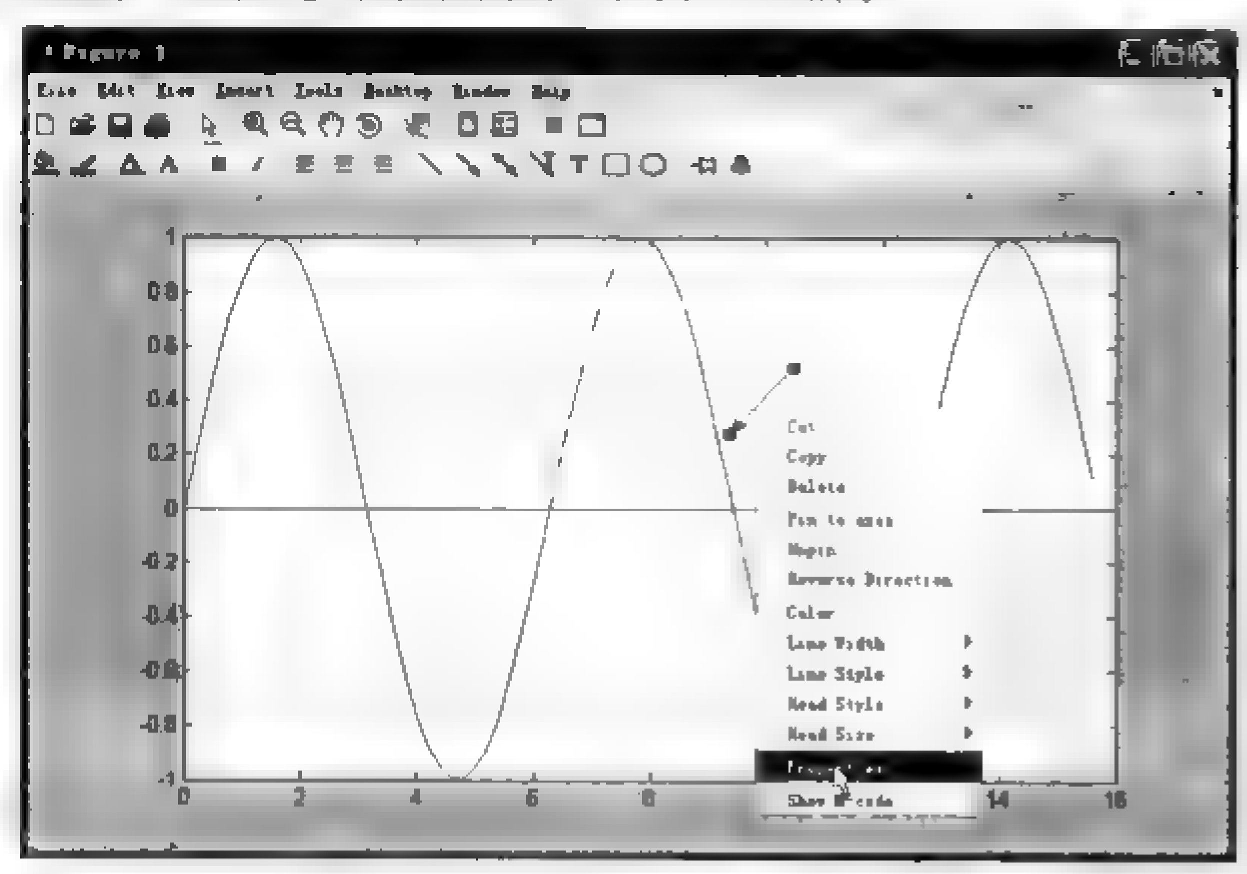


图 12-39　箭头和直线的编辑菜单的调出

单击 Properties...选项，将调出所选对象的属性框，用户可以对所选对象进行编辑，如图 12-40 所示。

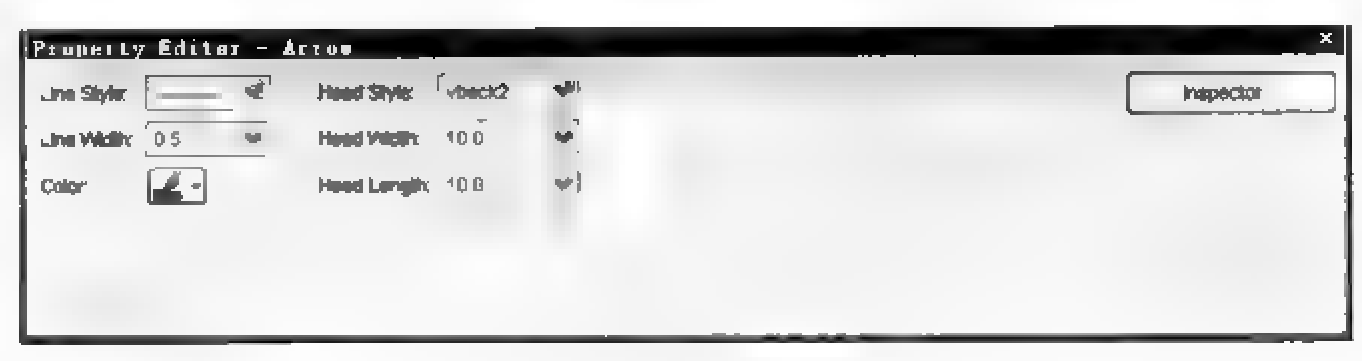


图 12-40　箭头的编辑

5. 图例的添加

用户在图形绘制过程中，经常会出现在同一图形窗口中绘制多条曲线的情况，为了更好地区分各条曲线，对它们表示的数据进行更准确的区分，可以使用图例加以说明。在 MATLAB 7.0 语言中，可以使用以下 3 种方法生成图例。

- 打开 Insert 菜单中并选择 Legend 命令；
- 单击工具栏中的 legend 图标；
- 使用 legend 函数。

下面对使用前两种方法进行图例添加做简单的介绍，其操作步骤如下。

(1) 单击工具栏中的 legend 图标，或是打开 Insert 菜单并选择 Legend 命令，如图 12-41 所示。

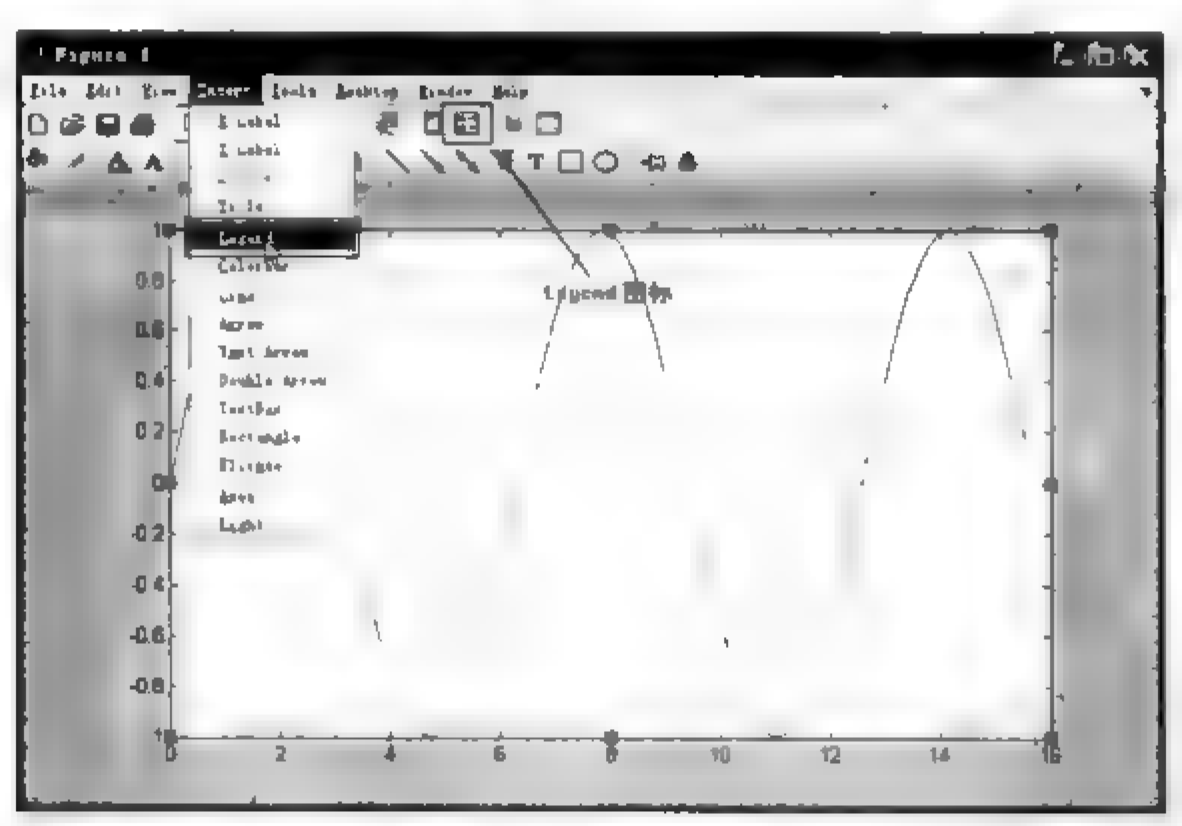


图 12-41　添加图例 1

(2) 此时，在图形窗口中自动生成图例，此时的图例文字为 “data1”，如图 12-42 所示。此时，用户可以双击该图例，修改图例中的文字。

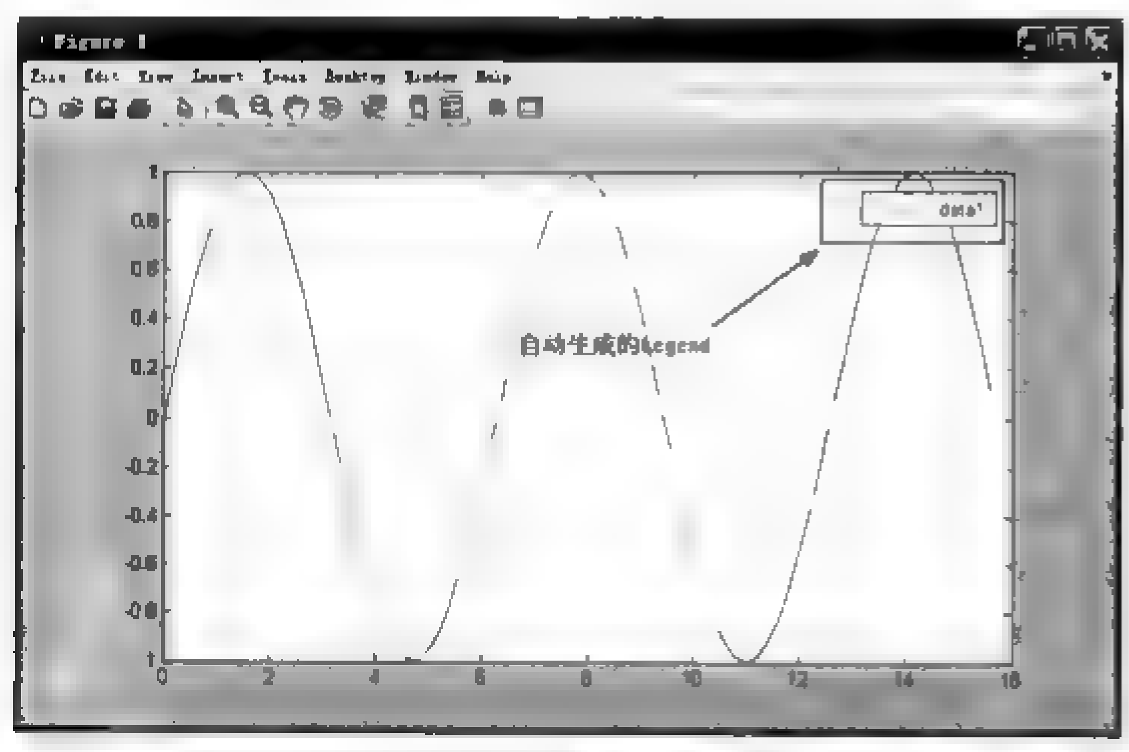


图 12-42 添加图例 2

(3) 修改完图例中的数据之后，用户可以按住鼠标左键将图例拖动到图形中的任意位置，如图 12-43 所示。

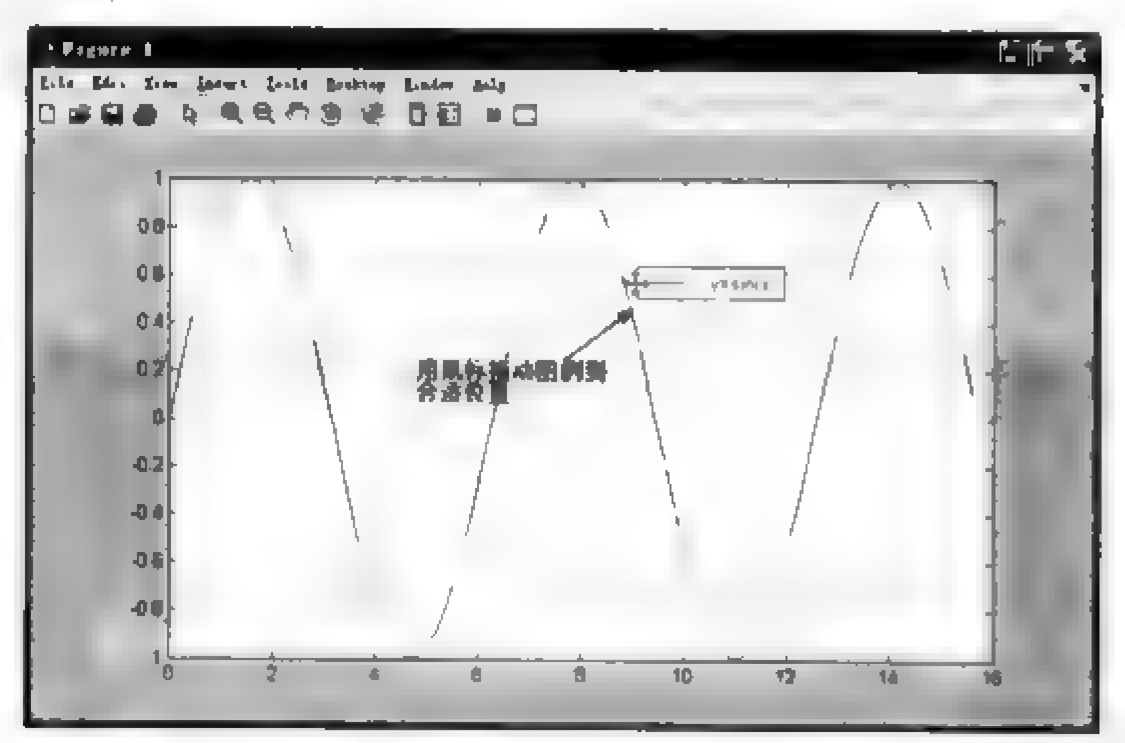


图 12-43 使用鼠标拖动图例到合适位置

同时，使用 legend 函数也可以进行图例标注。legend 函数对图形中的所有曲线进行的是自动标注，以输入的变量作为输入文本，它通常的调用形式如下。

- *legend*(*string*1,*string*2,*string*3,...) 命令在当前图形中输入标注语句 *string*1,*string*2,*string*3,...，标注的顺序对应绘图过程中按绘制先后顺序所生成的曲线。标注文本的大小和形式与相应坐标系的形式对应；
- *legend*(*H*,*string*1,*string*2,*string*3,...) 命令在当前图形中输入标注语句 *string*1,*string*2,*string*3,...，并且使用在向量 H 中定义的句柄，标注包含对应于相应句柄指定的文本；
- *legend off* 命令从当前坐标系中删除 legend 函数所生成的图例标注；
- *legend hide* 命令使得 legend 函数所生成的图例标注不可见；

- *legend show* 命令使得 legend 函数所生成的图例标注可见；
- *legend*(..., *Pos*) 命令将 legend 函数所生成的图例标注放置在指定的位置，其中 *Pos* 的取值和意义如下：

  0 = 系统自动定位，使得图形与标注重复最少，即最优化标注；

  1 = 置于图形的右上角(系统默认值)；

  2 = 置于图形的左上角；

  3 = 置于图形的左下角；

  4 = 置于图形的右下角；

  -1 = 置于图形的右外侧。

如果用户还想移动 legend 函数所生成的图例标注，只需在图形窗口中的图例标注上按下鼠标左键并拖动，就可以将其移动到需要的位置；如果用户想对图例标注的文本进行修改，则只需在图形窗口中的图例标注上双击鼠标左键，就可以进行相关修改操作。

例 12-18　使用 legend 函数进行图形的图例标注。

解：在命令窗口输入如下程序，并按 Enter 键确认输入，运行结果如图 12-44 所示。

```
>> % 本程序显示 LEGEND 函数的用法
>> % 用户可以先使用 PLOT 函数绘出图形，再使用 LEGEND 函数进行图形的图例标注
>> x = 0:.1:20;
>> plot(x,bessel(1,x),x,bessel(2,x),x,bessel(3,x));
>> legend('第一条','第二条','第三条');
>>
```

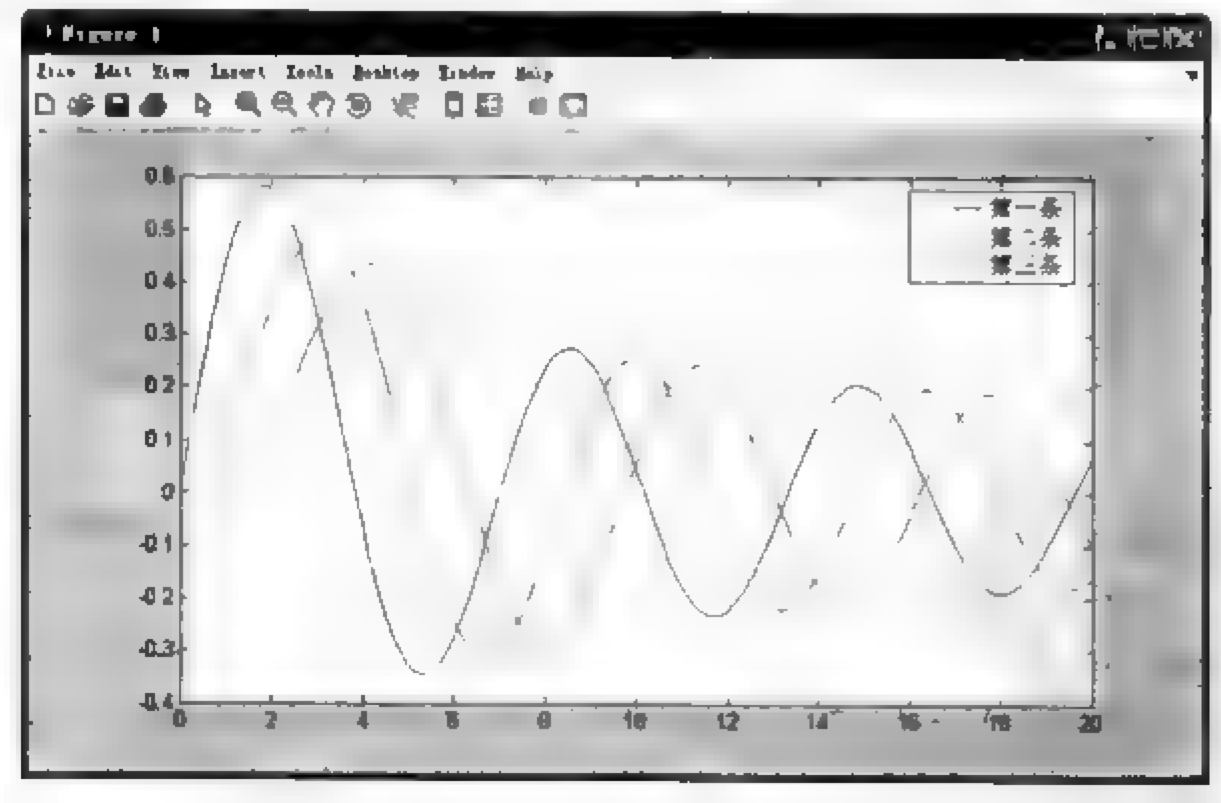


图 12-44　*legend*(*string*1, *string*2, *string*3,...) 命令

继续在命令窗口输入如下程序，并按 Enter 键确认输入，运行结果如图 12-45 所示。

```
>> legend('第一条','第二条','第三条',2);
>>
```

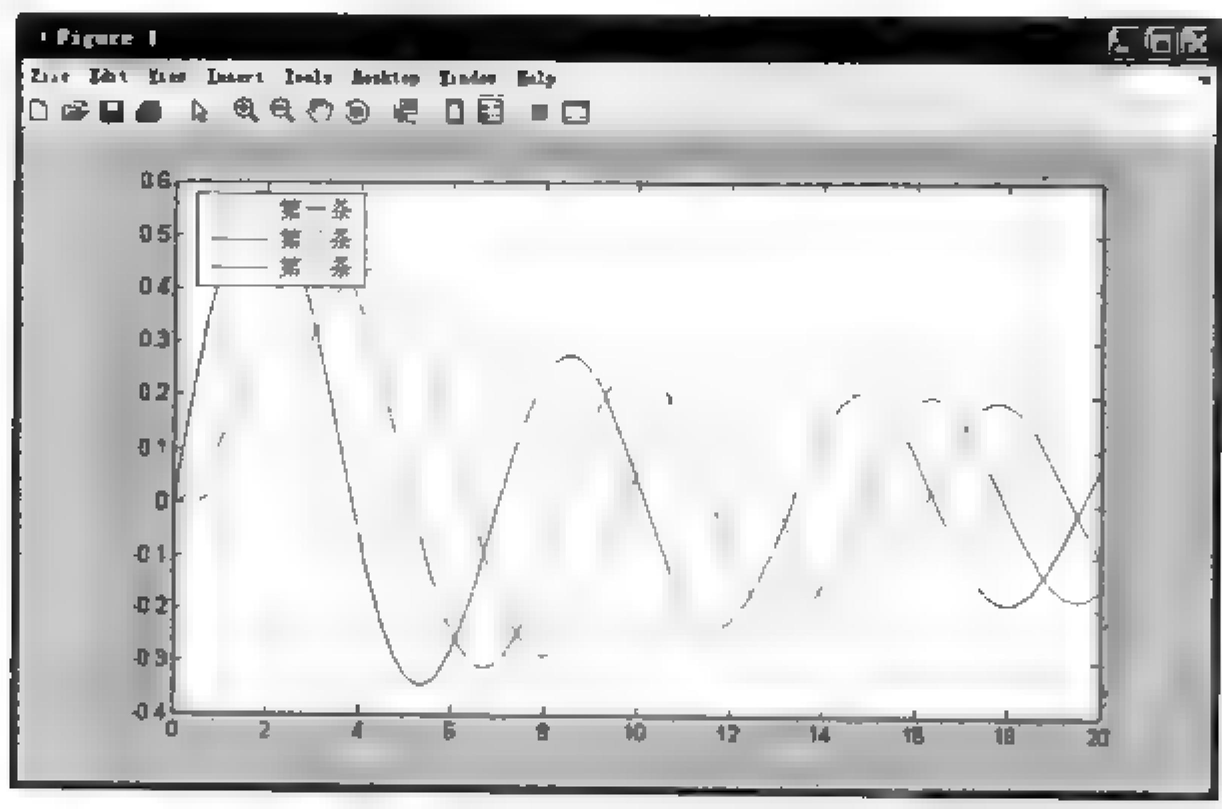


图 12-45　*legend*(..., *Pos*) 命令

在图形窗口中的图例标注上按下鼠标左键并拖动，将图例标注移动到需要的位置，结果如图 12-46 所示。

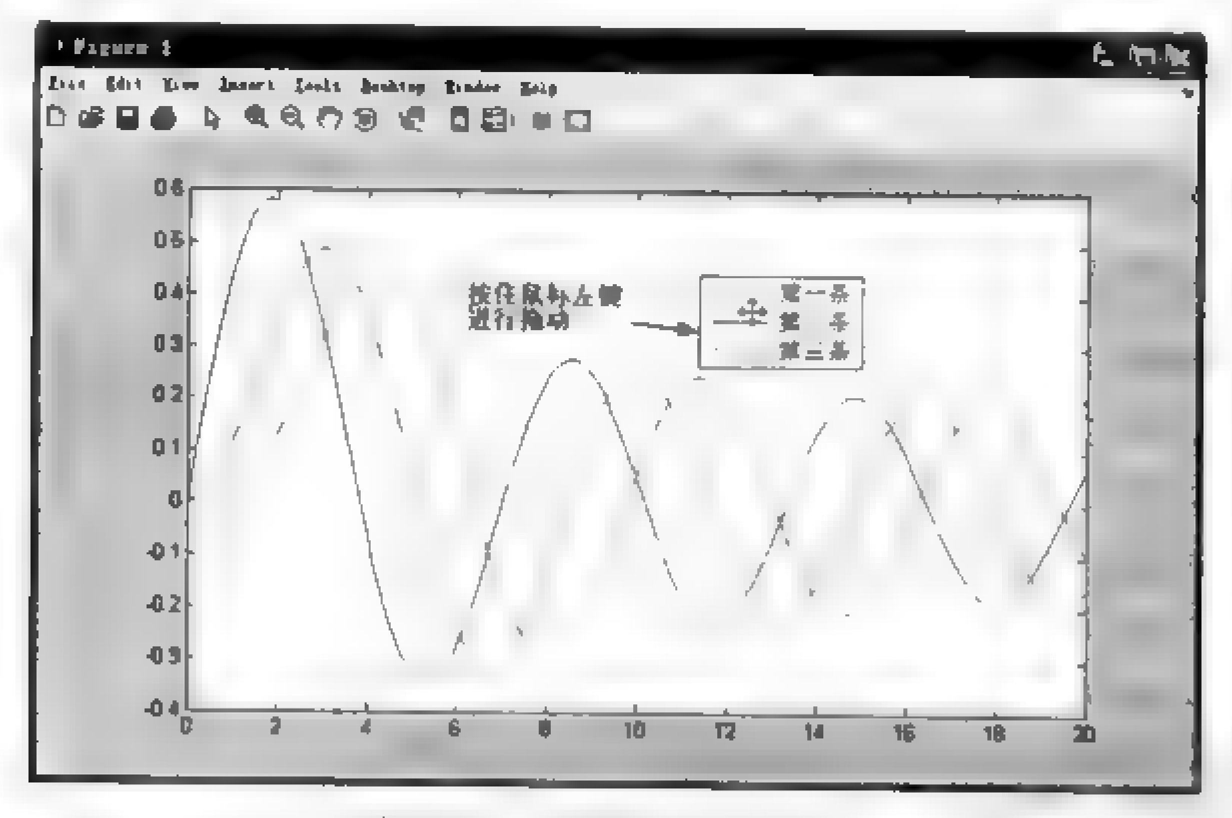


图 12-46　按下鼠标左键并拖动

下面再介绍一下怎样对图例进行编辑，右击图例，在弹出的菜单中选择 Properties 选项，MATLAB 7.0 将在图形窗口的下边弹出一个 Legend 的属性编辑器，如图 12-47 所示。

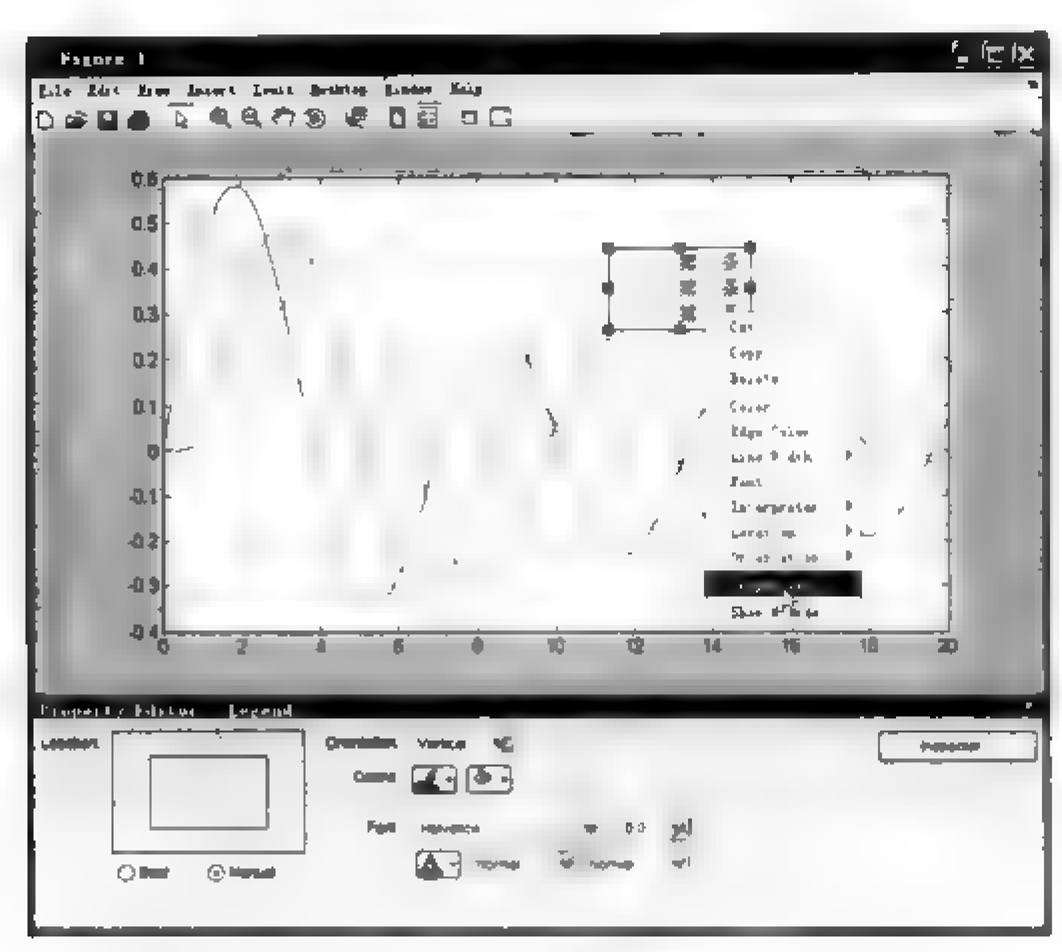


图 12-47　图例的编辑

用户可以通过修改 Legend 属性编辑器中的数据来对图例进行编辑，例如，改变 Color 的值可以改变图例的背景颜色；改变 Edge Color 的值可以改变 legend 边框的颜色；改变 Line Width 的值可以改变边框的线宽等。

6. 坐标网格的添加

在图形绘制过程中，为了精确地知道图形上某点的坐标，需要绘制坐标网格来定位，MATLAB 7.0 语言中提供了 grid 函数来实现这一功能，这样很大地提高了图形的显示效果。它通常的调用形式如下。

- *grid off* 命令关闭坐标网格；
- *grid on* 命令打开坐标网格；
- *grid Minor* 命令使用更细化的网格；
- *grid*(*AX*,...)命令使用 AX 坐标系代替当前坐标系。

例 12-19　使用 grid 函数进行图形网格显示控制。

解：在命令窗口输入如下程序，并按 Enter 键确认输入，运行结果如图 12-48 所示。

```
>> % 本程序显示 GRID 函数的用法
>> % 用户可以先使用 plot 函数绘出图形，再使用 GRID 函数进行图形网格显示控制
>> x=0:0.01:2;
>> y=exp(x);
>> plot(x,y)
>> grid on
>>
```

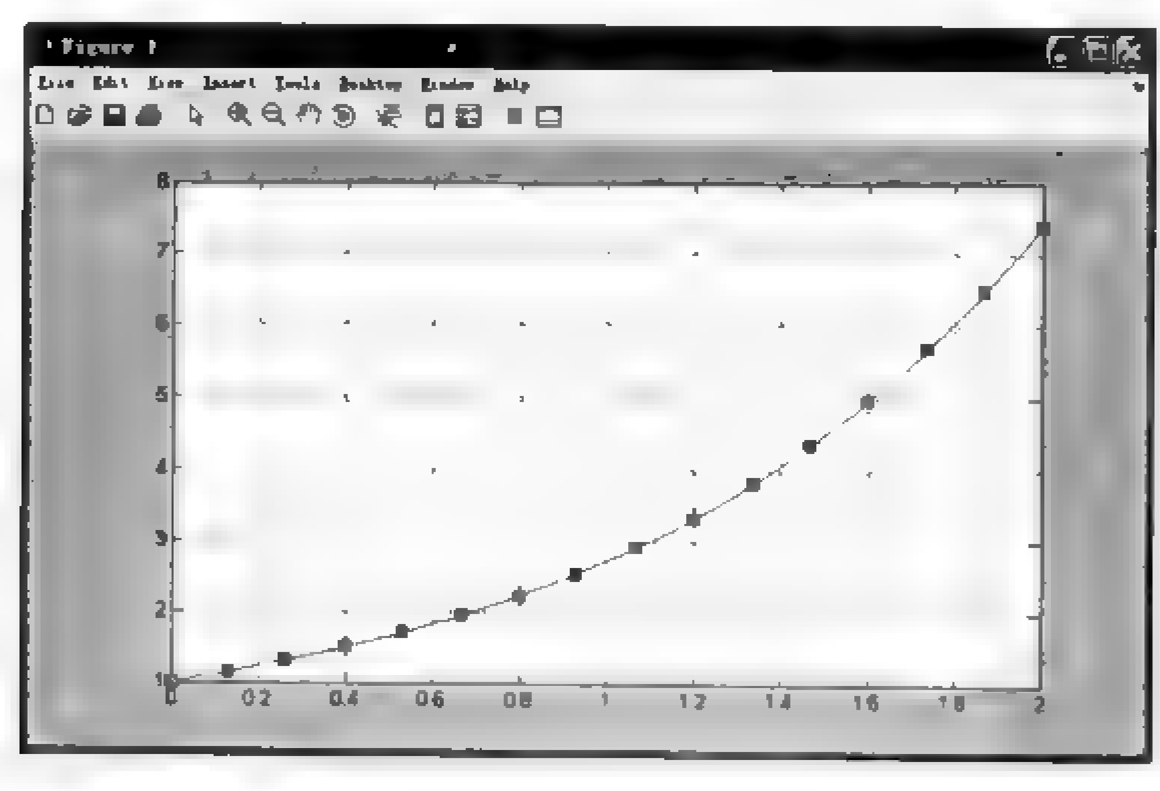


图 12-48　*grid on* 命令

继续在命令窗口输入如下程序，并按 Enter 键确认输入，运行结果如图 12-49 所示。

```
>> grid minor
>>
```

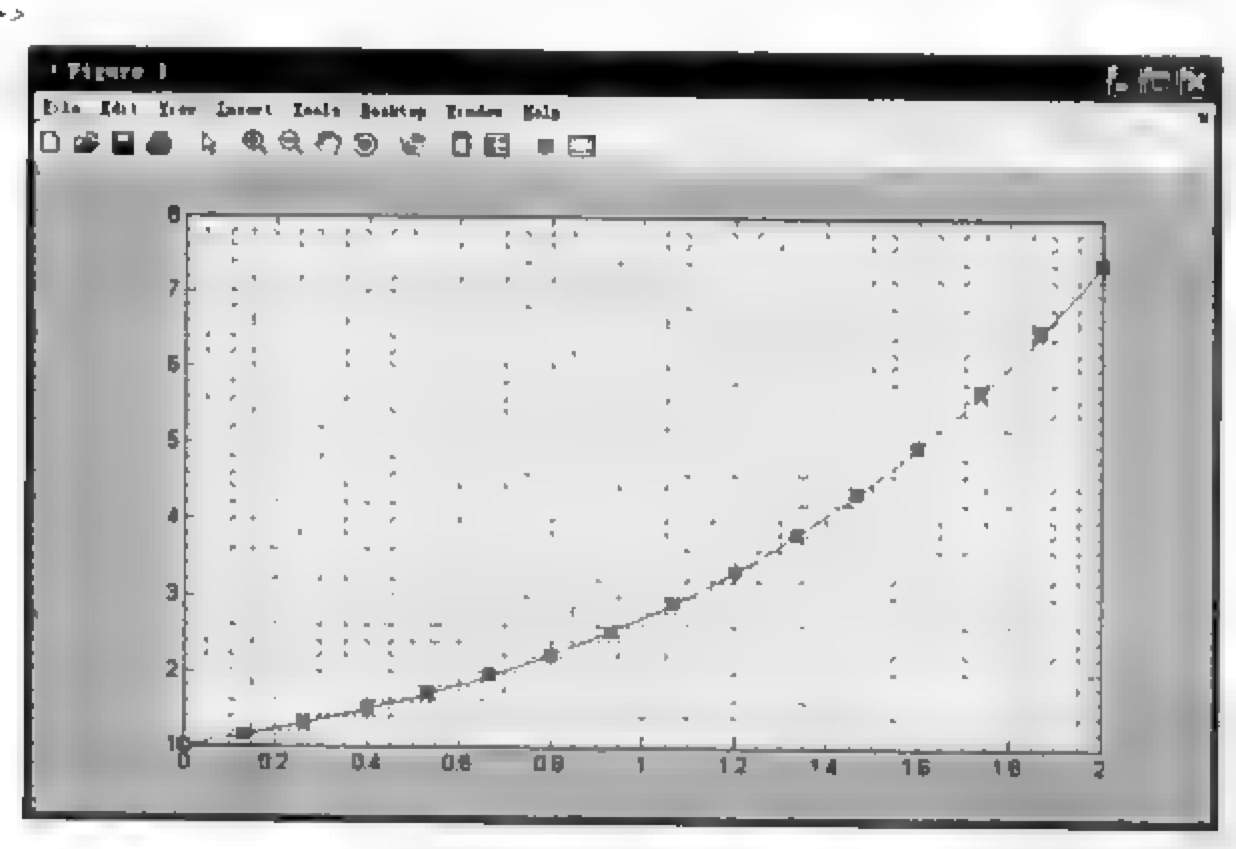


图 12-49　*grid Minor* 命令

7. 使用矩形或是椭圆在图形中圈出重要部分

用户可以使用矩形或是椭圆在图形中圈出特别的区域，从而使得该区域能引起特别的注意。当其中的一个矩形或是椭圆被选中时，用户可以移动并改变它的大小，或是右击它，在弹出的快捷菜单中用户可以选择改变它的属性和外观。

用户单击图形编辑工具栏中的矩形和椭圆图标即可在图形窗口中添加对应的图形，或是打开 Insert 菜单，从中选择 Rectangle 或是 Ellipse 命令，当鼠标形状改变成十字形时，

用户可以按下鼠标左键并拖动，当松开鼠标左键时，矩形或是椭圆的形状和大小就已经确定，如图 12-50 所示。

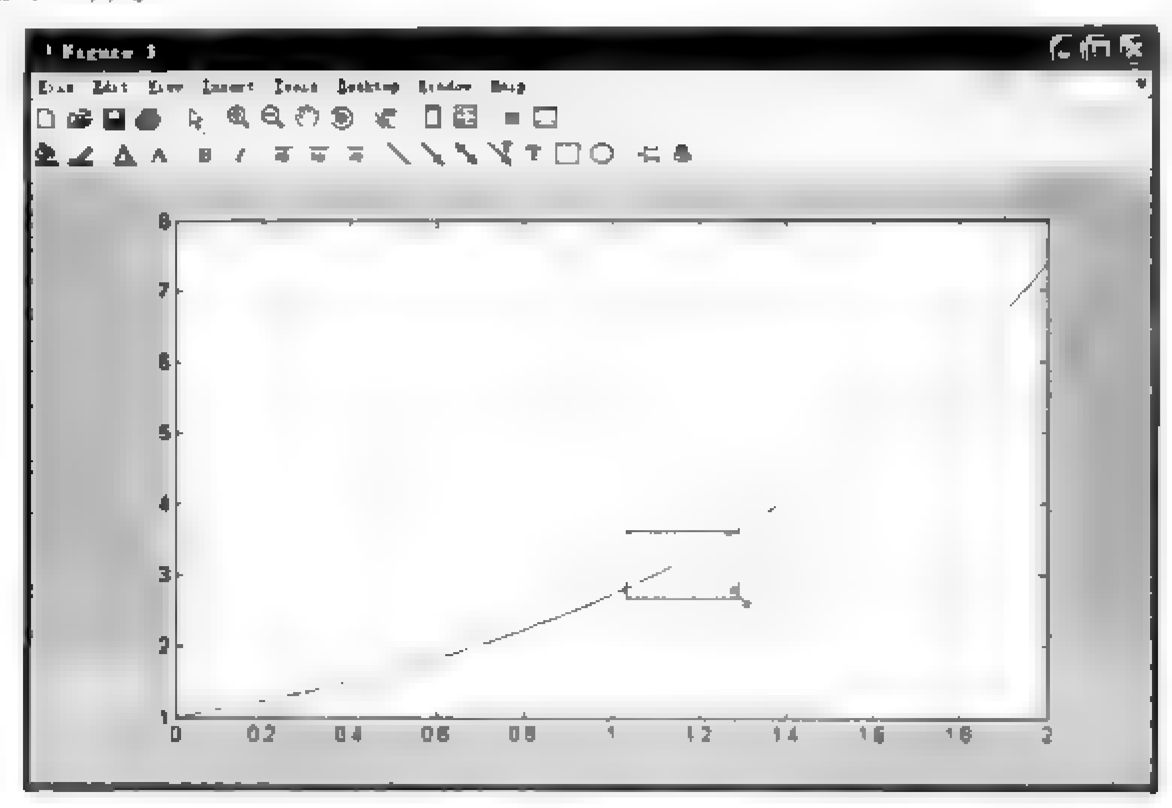


图 12-50　绘制矩形

当右击矩形或是椭圆时，将弹出一个菜单，如图 12-51 所示。

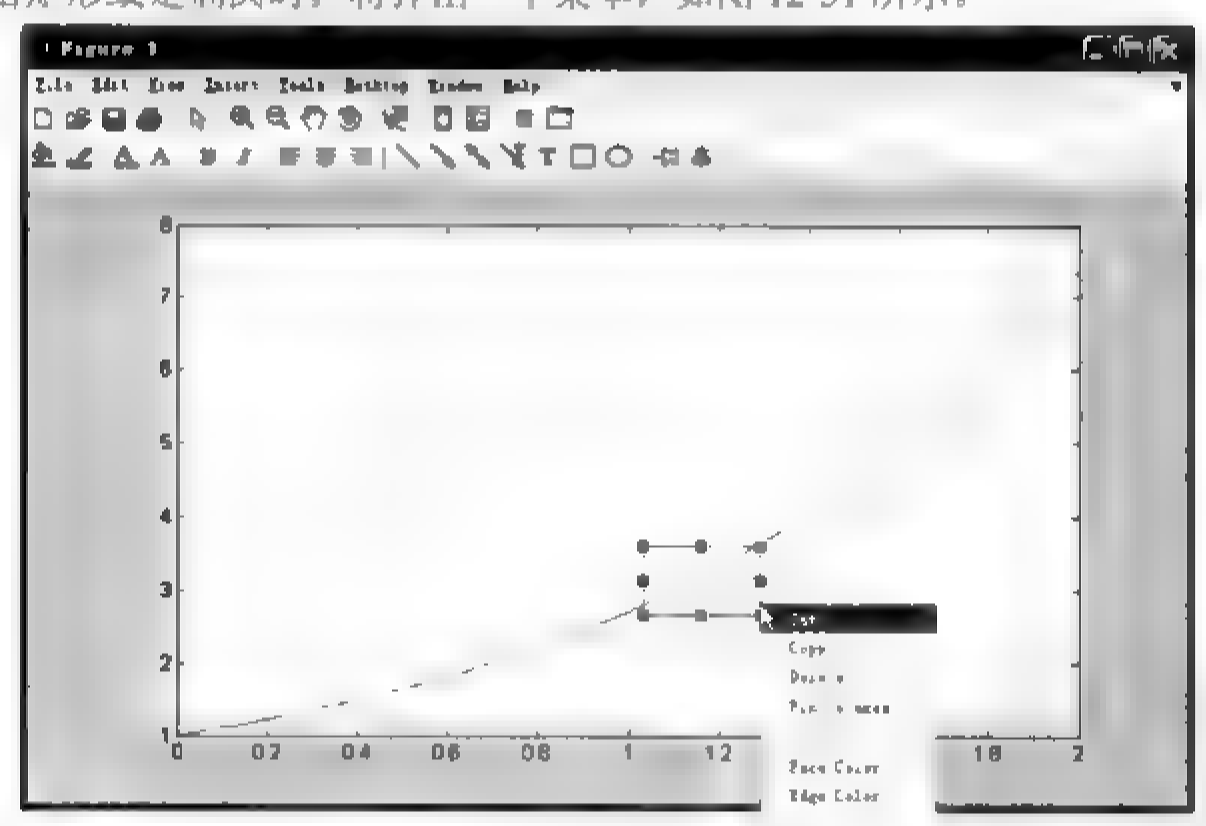


图 12-51　矩形的快捷菜单

该菜单包括如下命令。

- Cut、Copy 和 Delete：实现剪切、复制和粘贴功能。
- Pin to axes：将图形的左下角定位于当前位置。
- Unpin：将矩形在配属点分开。
- Face Color：给矩形或椭圆填充颜色。
- Edge Color：定义矩形或椭圆的颜色。
- Line Width：定义用于绘制矩形或是椭圆的颜色

- Line Style：定义用于绘制矩形或椭圆的线型。
- Properties：显示属性编辑器，如图 12-52 所示。

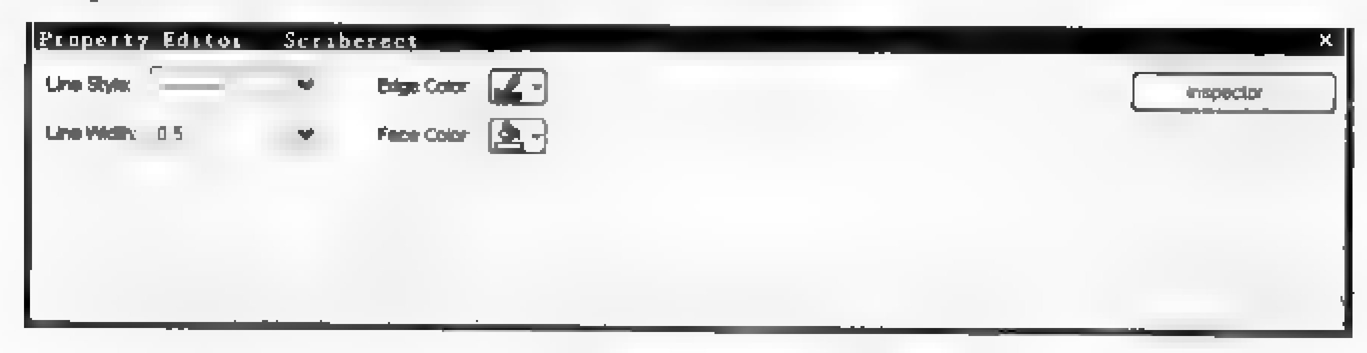


图 12-52 矩形或椭圆的属性编辑器

- Show M-code：创建再造图形的 M 文件代码，如图 12-53 所示。

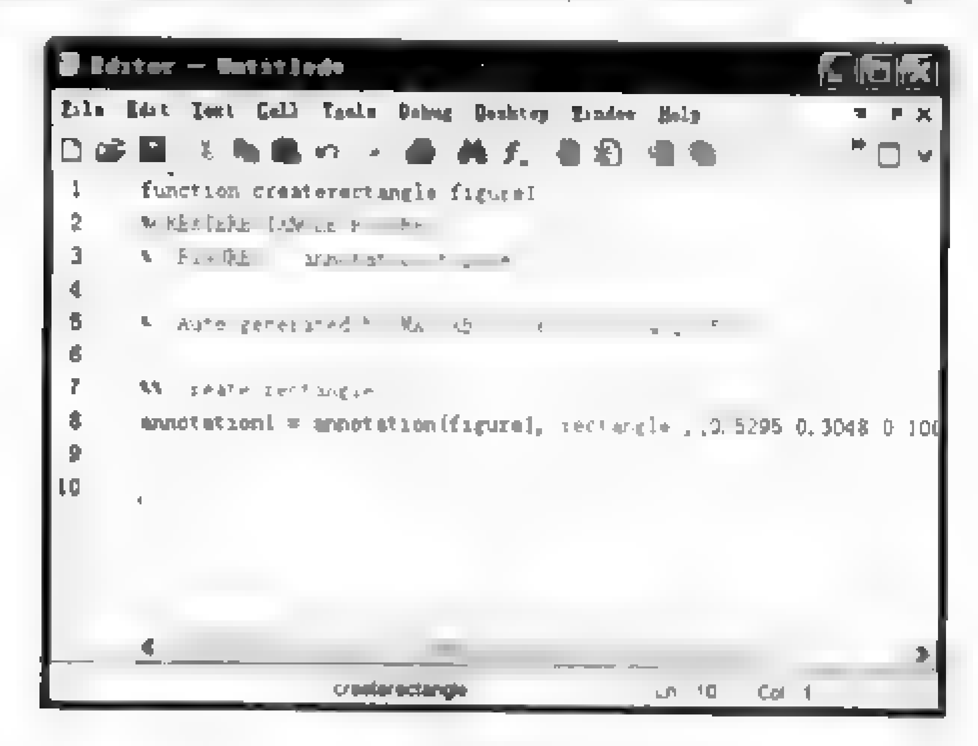


图 12-53 创建矩形的 M 文件

12.1.4 特殊图形的绘制

1. 条形图和面积图(Bar and Area Graphs)

条形图和面积图用于绘制向量和矩阵数据，这两种图形可以用来比较不同组数据的在总体数据中所占的比例，其中条形图适于表现离散型数据，而面积图适于表现连续型数据。

MATLAB 7.0 提供了以下一些函数来绘制各种条形图和面积图，如表 12-5 所示。

表 12-5 条形图和面积图的函数

函　数	功 能 描 述
bar	绘制矩阵 Y(m×n)各列的垂直条形图，各条以垂直方向显示
barh	绘制矩阵 Y(m×n)各列的垂直条形图，各条以水平方式显示
bar3	绘制矩阵 Y(m×n)各列的三维垂直条形图，各条以垂直方向显示
bar3h	绘制矩阵 Y(m×n)各列的三维垂直条形图，各条以水平方式显示
area	绘制向量的堆栈面积图

上表中，前 4 个函数用于绘制条形图，前两个函数用于绘制二维图形，后两个图形用于绘制三维图形，而第 1 个和第 3 个绘制的是垂直条形图，第 2 个和第 4 个绘制的是水平条形图。

下面对这些函数的使用做简单介绍。

(1) 集合式条形图

在默认情况下，矩阵中的每一个元素在图形中都有一个条形来代表，在二维条形图中，所有的条形都沿 x 轴分布，其中每一行的元素集中在 x 轴的某一个地方。

例 12-20　集合式条形图的绘制。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> Y = [5 2 1
      3 1 4
      1 5 9
      5 5 5
      4 3 2];
>> bar(Y)
>>
```

生成的图形如图 12-54 所示。

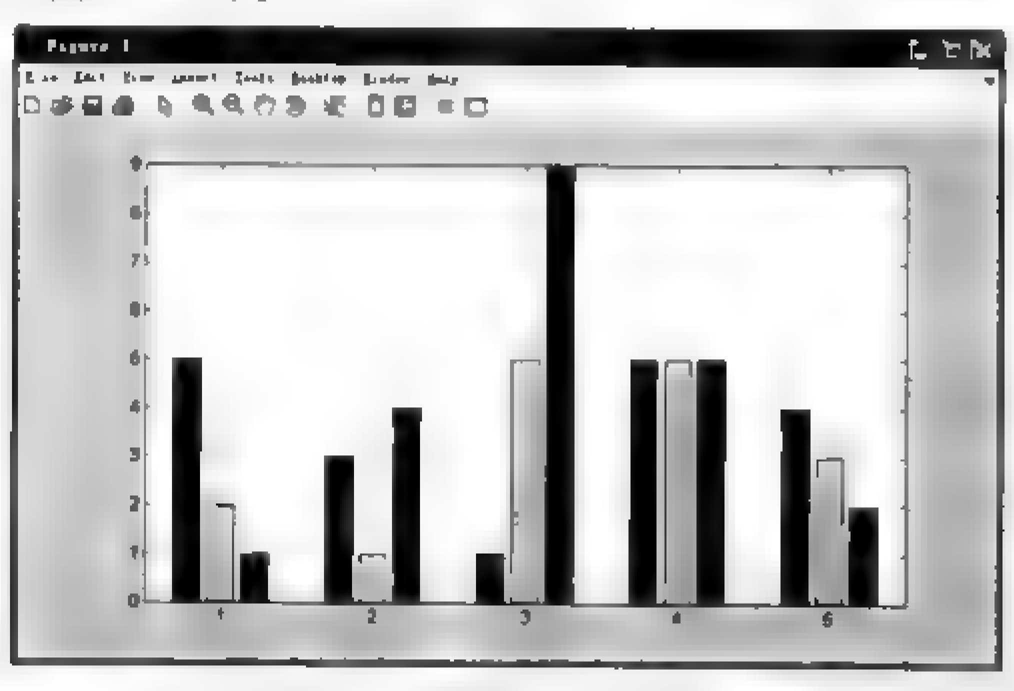


图 12-54　集合式条形图的绘制

(2) 分离式三维条形图

函数 bar3 最简单的使用形式是将每一个元素以分离的三维条的形式表现出来，将每一列的元素沿 y 轴分布，其中第 1 列的元素以 x 轴的 1 为中心分布。

例 12-21　分离式三维条形图的绘制。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> Y = [5 2 1
      3 1 4
      1 5 9
      5 5 5
```

```
        4 3 2];
>> bar3(Y)
>>
```

生成的图形如图 12-55 所示。

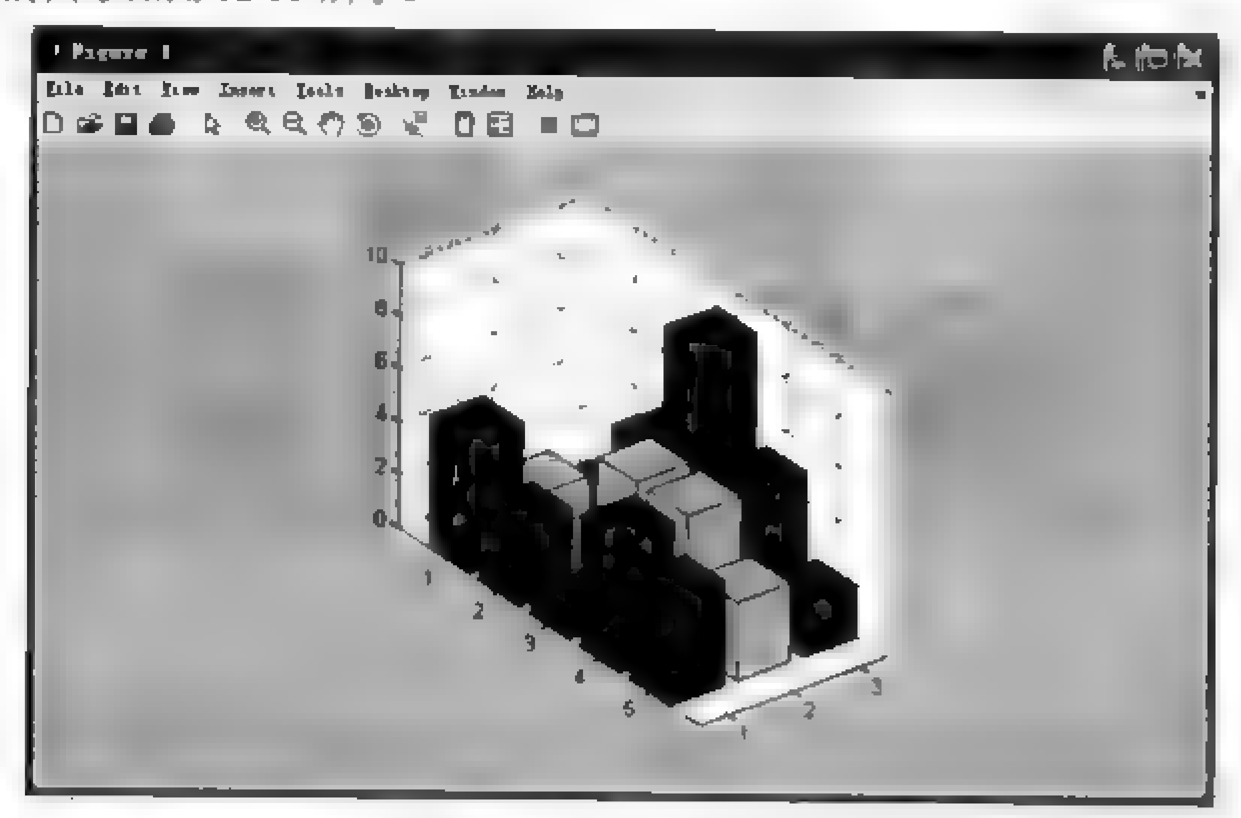


图 12-55　分离式三维条形图

此外，还可以绘制集合式三维条形图，使用格式为 bar3(Y,'group')，详细情况可以参见 MATLAB 7.0 的帮助系统。

(3) 堆叠式条形图

堆叠式条形图可以显示矩阵中各个元素在其所在行中所占的比例，这种图形将矩阵中的每一行以一个条形显示，每一条被分隔成 n 个区段，其中 n 为矩阵每列的元素数目，对于垂直条形图，每一个条形的高度等于每行元素的总和，而条形中的每一段为对应元素的值。

例 12-22　堆叠式条形图的绘制。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> Y = [5 1 2
        8 3 7
        9 6 8
        5 5 5
        4 2 3];
>> bar(Y,'stack')
>> grid on
>> set(gca,'Layer','top')
>>
```

所得图形如图 12-56 所示。

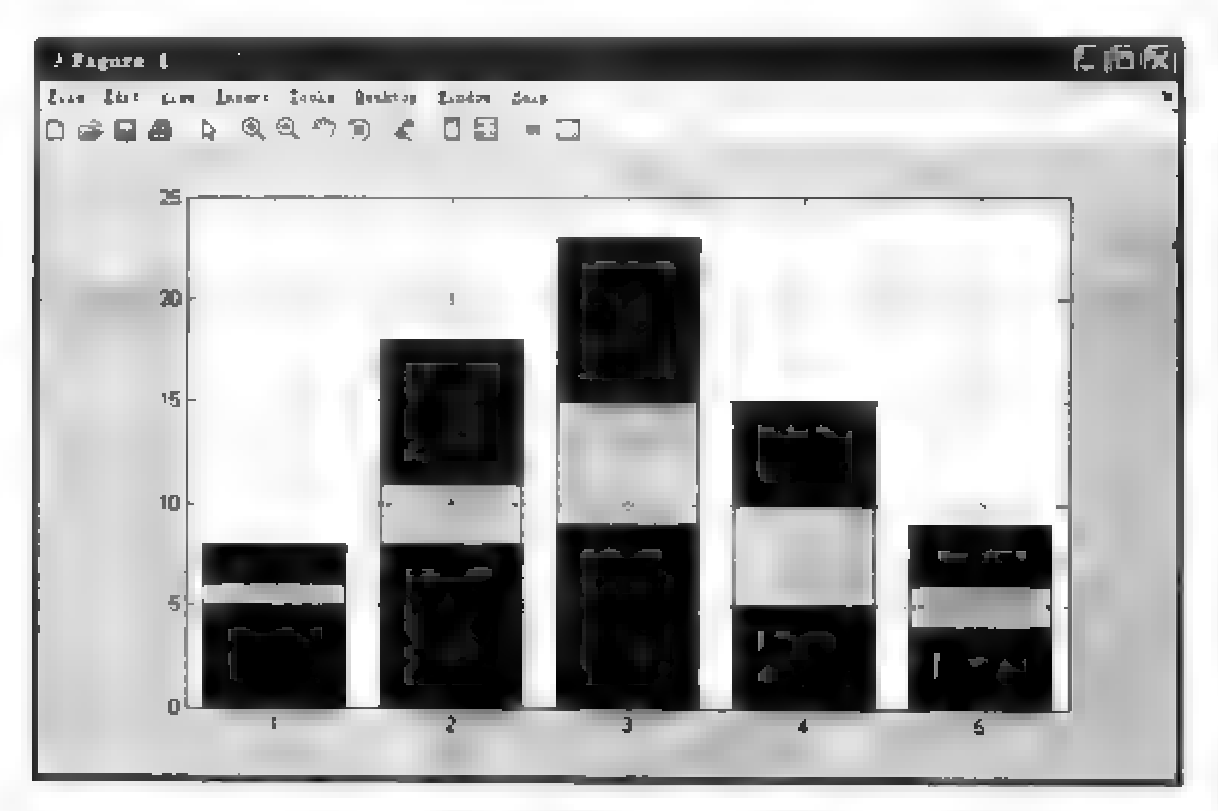


图 12-56　堆叠式条形图

(4) 面积图

函数 area 显示向量或矩阵中各列元素的曲线图，该函数将矩阵中的每列元素分布绘制曲线，并填充曲线和 x 轴之间的空间。

面积图在显示向量或是矩阵中的元素在 x 轴每特定点占所有元素的比例时十分有效。在默认情况下，函数 area 将矩阵中各行的元素集中并将这些值绘成曲线。

仍以上例的矩阵 Y 为例，在命令窗口中输入如下命令，并按 Enter 键确认，如图 12-57 所示。

```
area(Y)
```

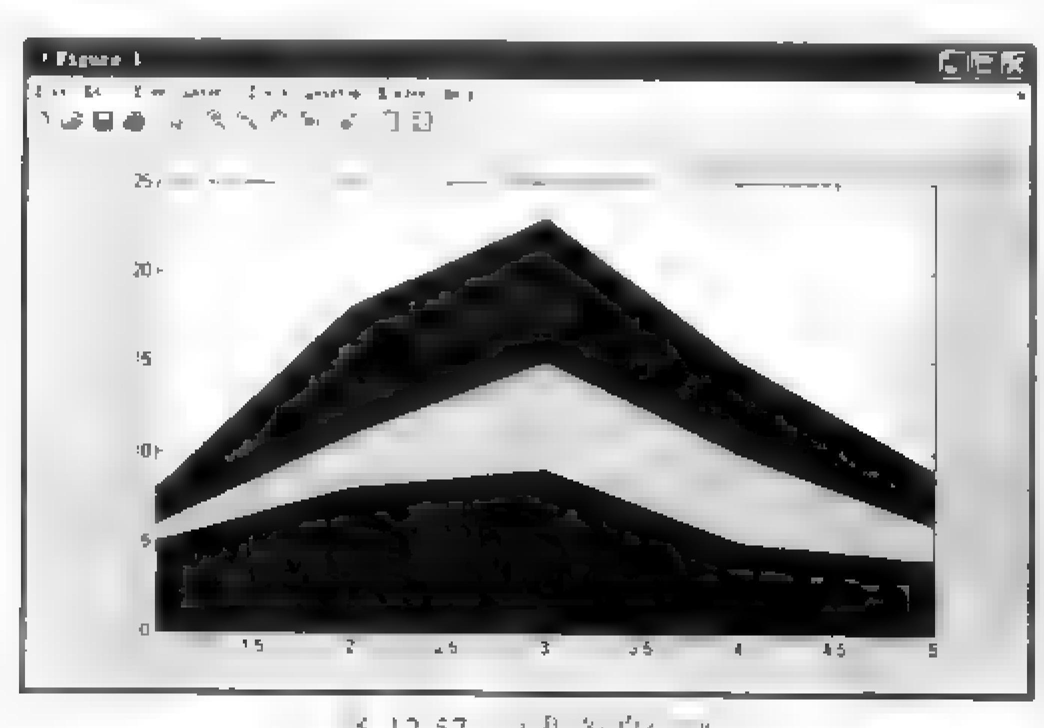


图 12-57　面积图的绘制

2. 饼形图(Pie Charts)

在统计学中，人们经常要用到饼形图来表示各个统计量占总量的份额，饼形图可以显示向量或矩阵中的元素占所有元素总和的百分比，MATLAB 提供了 pie 函数和 pie3 函数，分别用于绘制二维饼形图和三维饼形图。下面对这两个函数的用法做简单介绍。

- $pie(x)$ 命令和 $pie3(x)$ 命令绘制关于向量 x 的各个分量的饼形图，其中前者绘制二维图形，后者绘制的是三维图形。x 的各个分量先被除以 $sum(x)$，这样可以决定各个分量在图形中的“饼块份额”。如果 $sum(x) <= 1.0$，那么向量 x 各个分量的值将直接成为“饼块份额”。此时 MATLAB 7.0 将只绘出一个不完整的饼形图。
- $H = pie(x,e)$ 命令和 $H = pie3(x,e)$ 命令可以绘制出饼块分离的饼形图，其中前者绘制二维图形，后者绘制的是三维图形。向量 e 必须和 x 具有相同的维数。e 和 x 的分量一一对应，若其中有分量不为零，则 x 中的对应分量将被分离出饼形图。
- $H = pie(...,labels)$ 命令和 $H = pie3(...,labels)$ 命令可以给每一个饼块取名，向量 $labels$ 必须和 x 具有相同的维数且只能为字符串。其中前者绘制二维图形，后者绘制的是三维图形。

例 12-23　使用 pie 函数和 pie3 函数绘制饼形图。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>>% 以下程序用于绘制 SUM(X) <= 1.0 时的饼形图，并给各饼块加了标签
>> x=[0.1 0.2 0.3 0.1];
>> label={'North','South','East','West'};
>> pie(x,label)
>>
```

结果如图 12-58 所示。

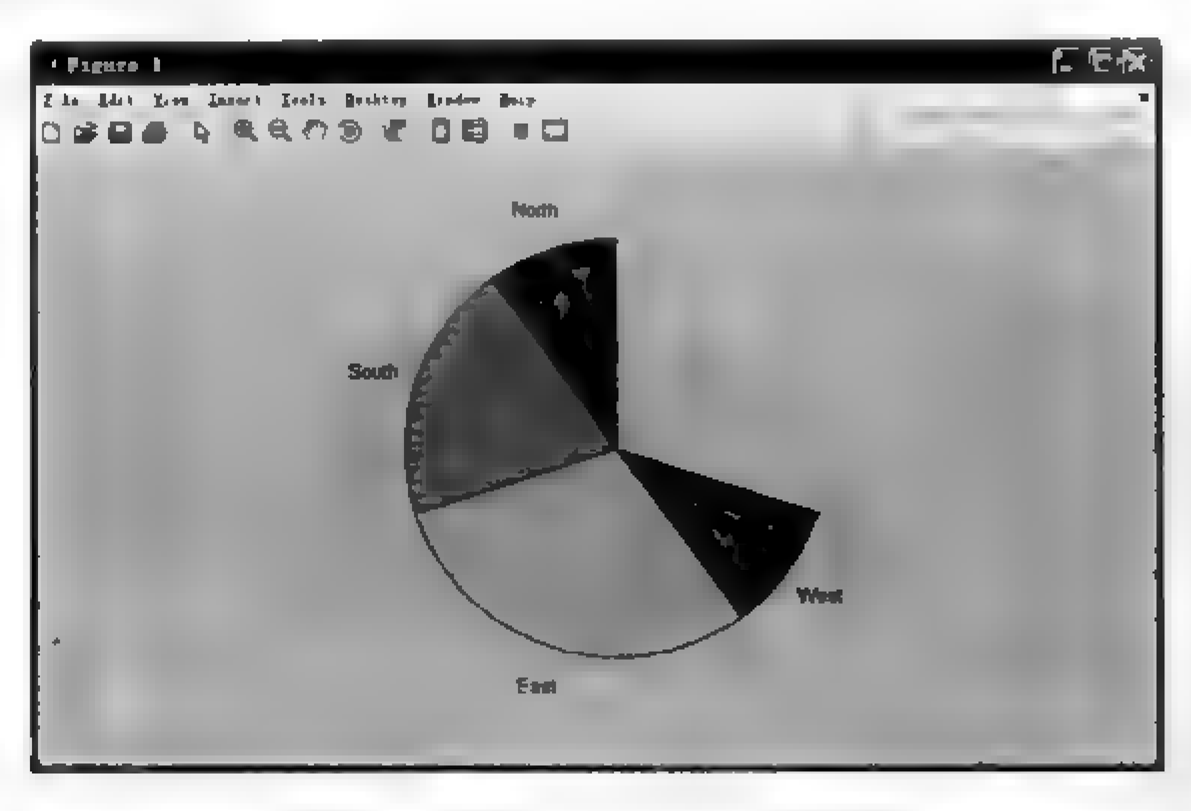


图 12-58　绘制 $sum(x) <= 1.0$ 时的饼形图

```
>>% 下边的这段程序用于绘制关于 5 个国家的人口比重的饼形图
```

```
>>% 我们使用 EXPLODE 参数将中国所代表的那一部分饼块从中单独绘出。
>> X=[13 10 3 2 1];
>> EXPLODE=[1 0 0 0 0];
>> label={'China','India','American','Japan','Russian'};
>> PIE(X,EXPLODE,label)
>>
```

运行结果如图 12-59 所示。

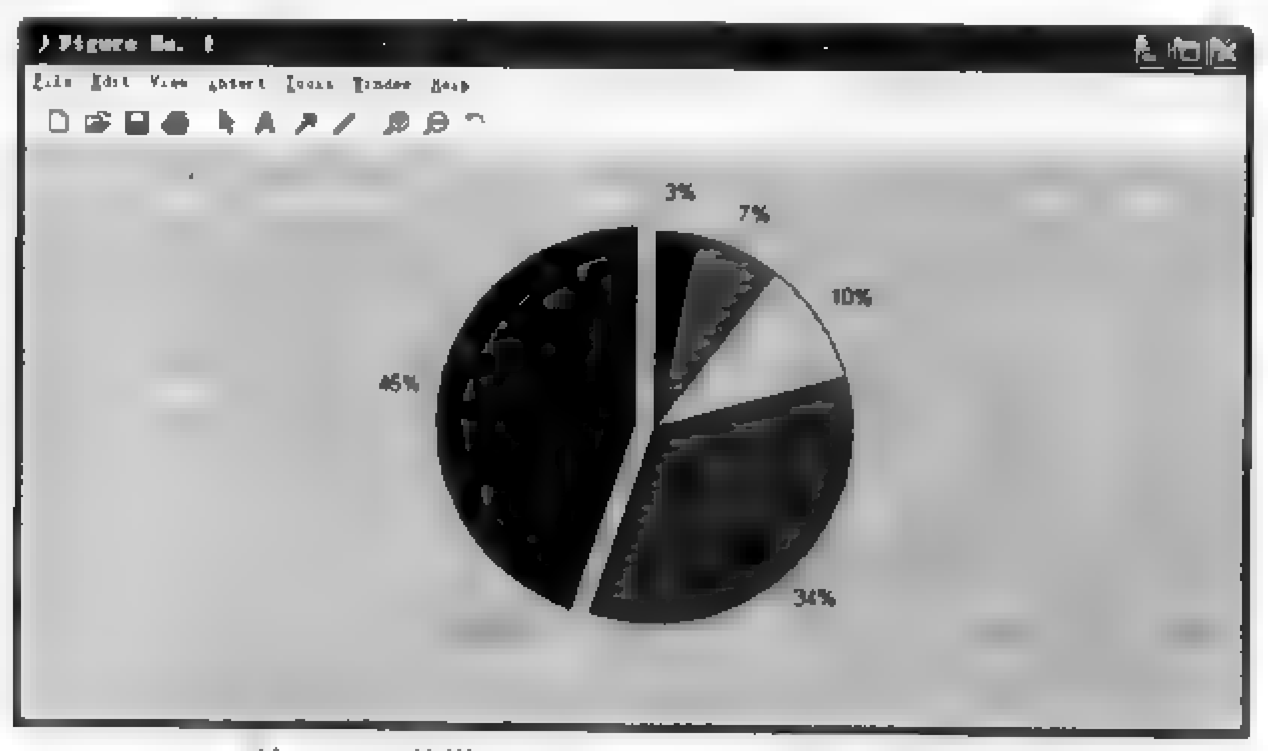


图 12-59　使用 $H = pie(...,labels)$ 命令绘制饼形图

```
>>% 以下程序用于绘制 SUM(X) <= 1.0 时的饼形图，并给各饼块加了标签
>> pie3([49 21 16 14],[1 1 1 0],{'Tsinghua','Peiking','Fudan','Zhejiang'})
>> colormap(cool)
>>
```

运行结果形如图 12-60 所示。

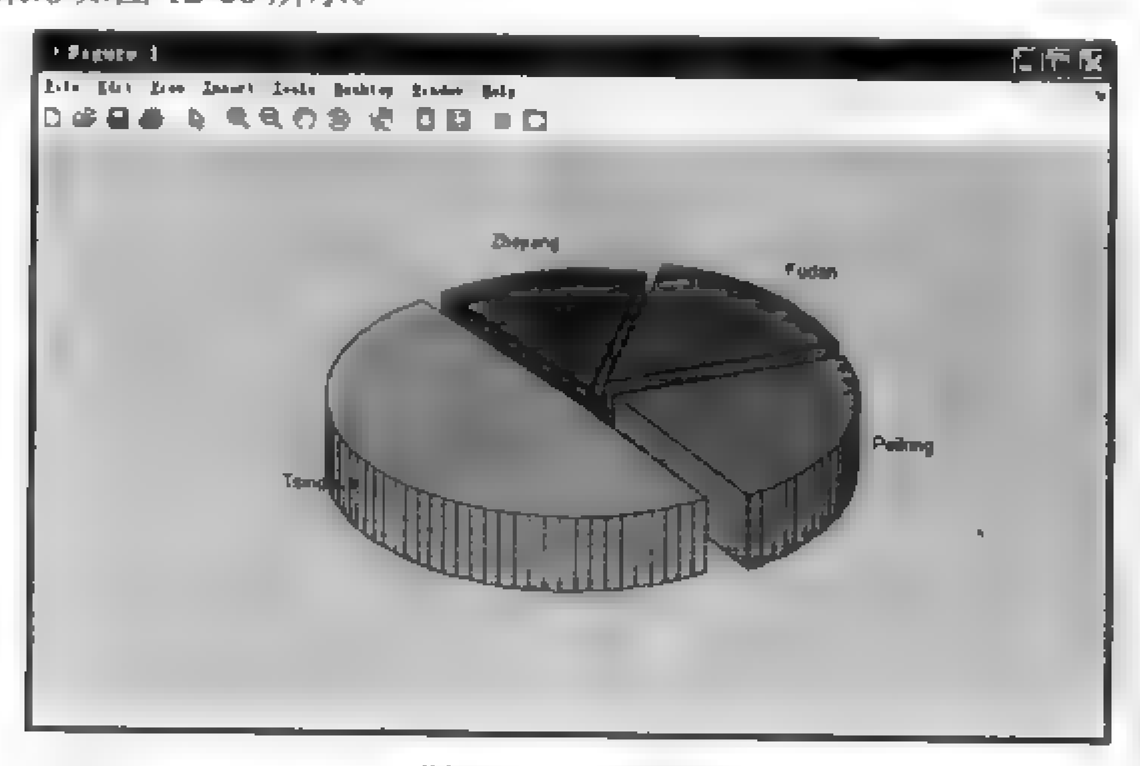


图 12-60　pie3 函数

3. 离散型数据图

MATLAB 7.0提供了一系列适于表现离散型数据的函数，本小节主要讲述怎样使用stem函数、stem3函数和stairs函数来显示这些数据(前文所述的条形图也适于表现离散型数据)。

它们的使用格式如下。

- *stem*(Y)命令可以绘制Y的数据序列，图形从起始于X轴，并在每个数据点处绘制圆圈；
- *stem*(X,Y)命令可以按照指定的X绘制数据序列Y；
- *stem3*(Z)命令绘制Z的数据序列，图形起始于X-Y平面，并在每个数据点处绘制圆圈；
- *stem3*(X,Y,Z)命令可以按照指定的X和Y值绘制数据序列Z。
- *stairs*(Y)命令按照向量Y的元素绘制出阶梯状图形；
- *stairs*(X,Y)命令按照指定X对应的向量Y中的元素绘制出梯状图形，需要注意的是X必须为单调递增。

下面对它们的使用方法举例予以说明。

例12-24　使用*stem*(Y)和*stem*(X,Y)命令绘制句柄状图形。

解：在命令窗口输入如下程序，并按Enter键确认输入，运行结果如图12-61和图12-62所示。

```
>> % 该程序用于绘制句柄状图形
>> % 用户先输入向量x和y的值，再输入stemv(y)可得运行结果
>> y=randn(40,1);
>> stem(y)
>>
```

以上程序运行之后所得图形如图12-61所示。

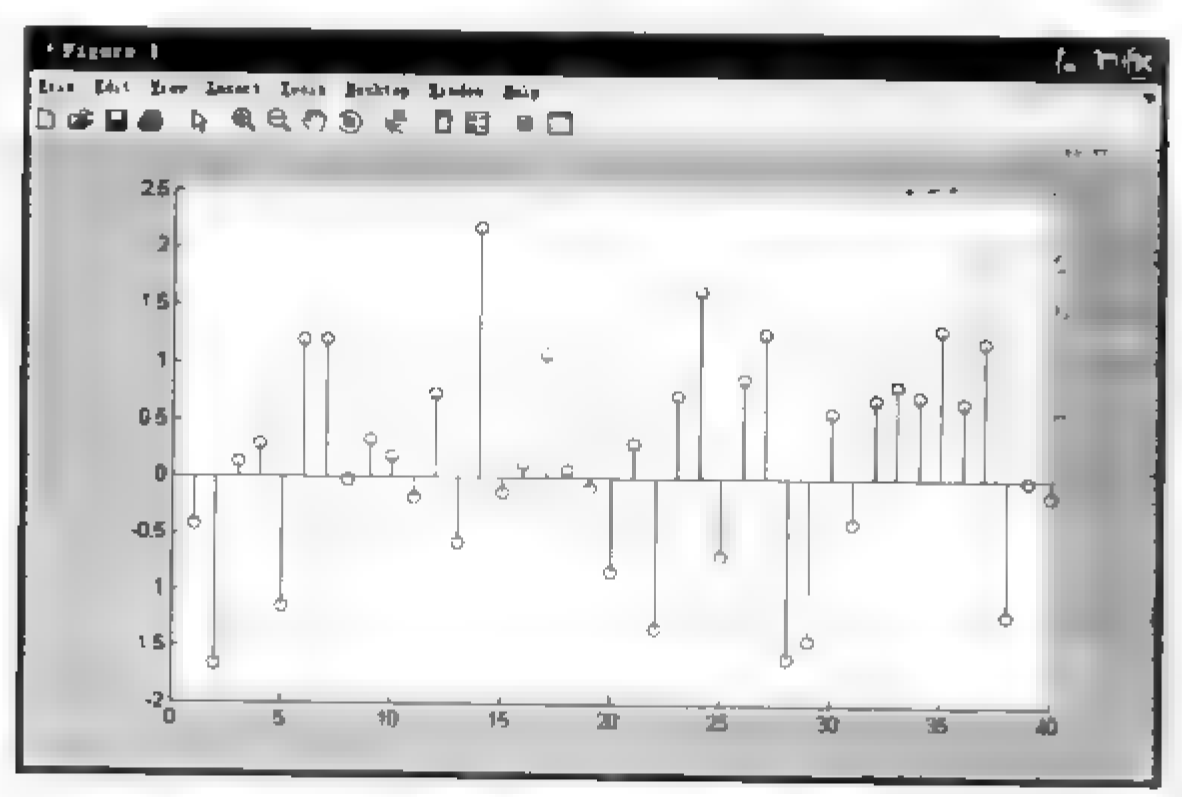


图12-61　使用*stem*(Y)命令绘制句柄状图形

```
>> % 该程序用于绘制句柄状图形
>> % 用户先输入向量 x 和 y 的值，再输入 stem(x,y,s)可得运行结果
>> % 这里将 s 定义为'r'，所以图形为红色
>> x=-10:0.25:10;
>> y=x.^2+2.*x;
>> stem(x,y,'r')
>>
```

以上程序运行之后所得图形如图 12-62 所示。

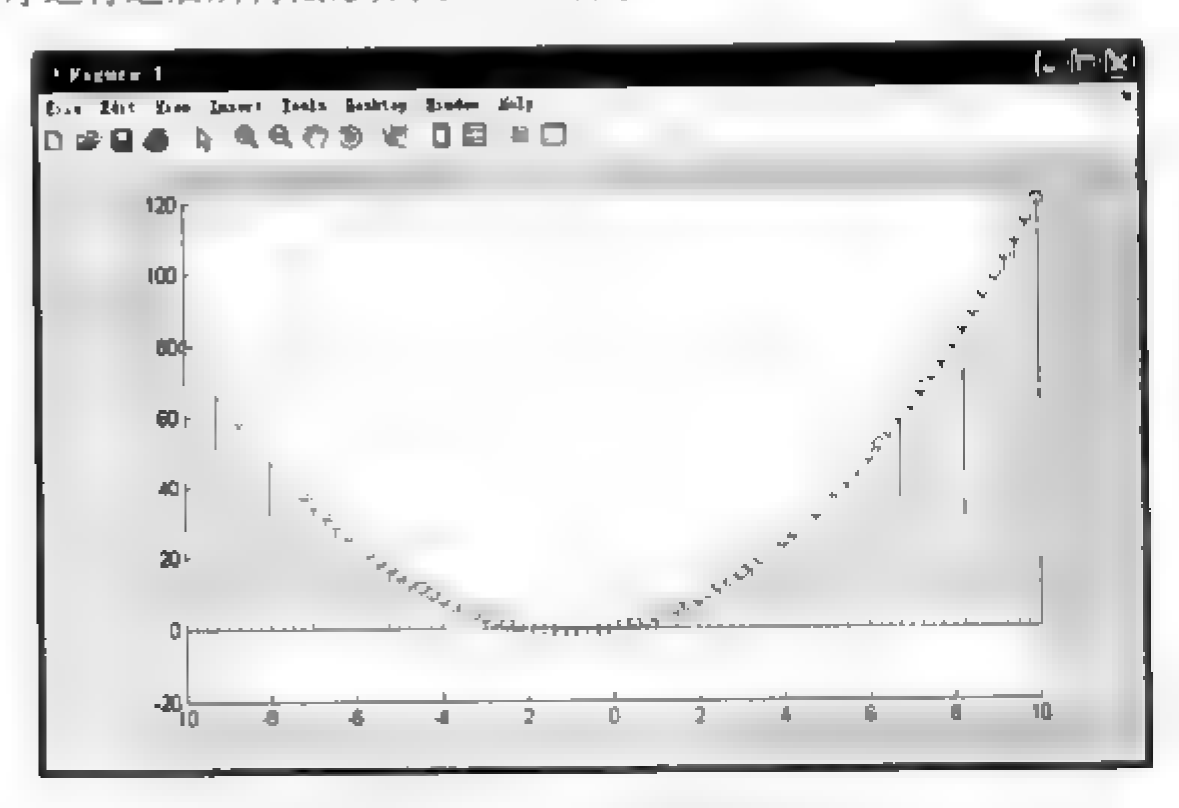


图 12-62　stem(X,Y) 命令绘制句柄状图形

例 12-25　使用 stem3 函数进行三维茎状图的绘制。

解：在命令窗口输入如下程序，并按 Enter 键确认。

```
>> th = (0:127)/128*2*pi;
>> x = cos(th);
>> y = sin(th);
>> f = abs(fft(ones(10,1),128));
>> stem3(x,y,f','d','fill')
>> xlabel('Real')
>>ylabel('Imaginary')
>>zlabel('Amplitude')
>>title('Magnitude Frequency Response')
>>
```

运行结果如图 12-63 所示。

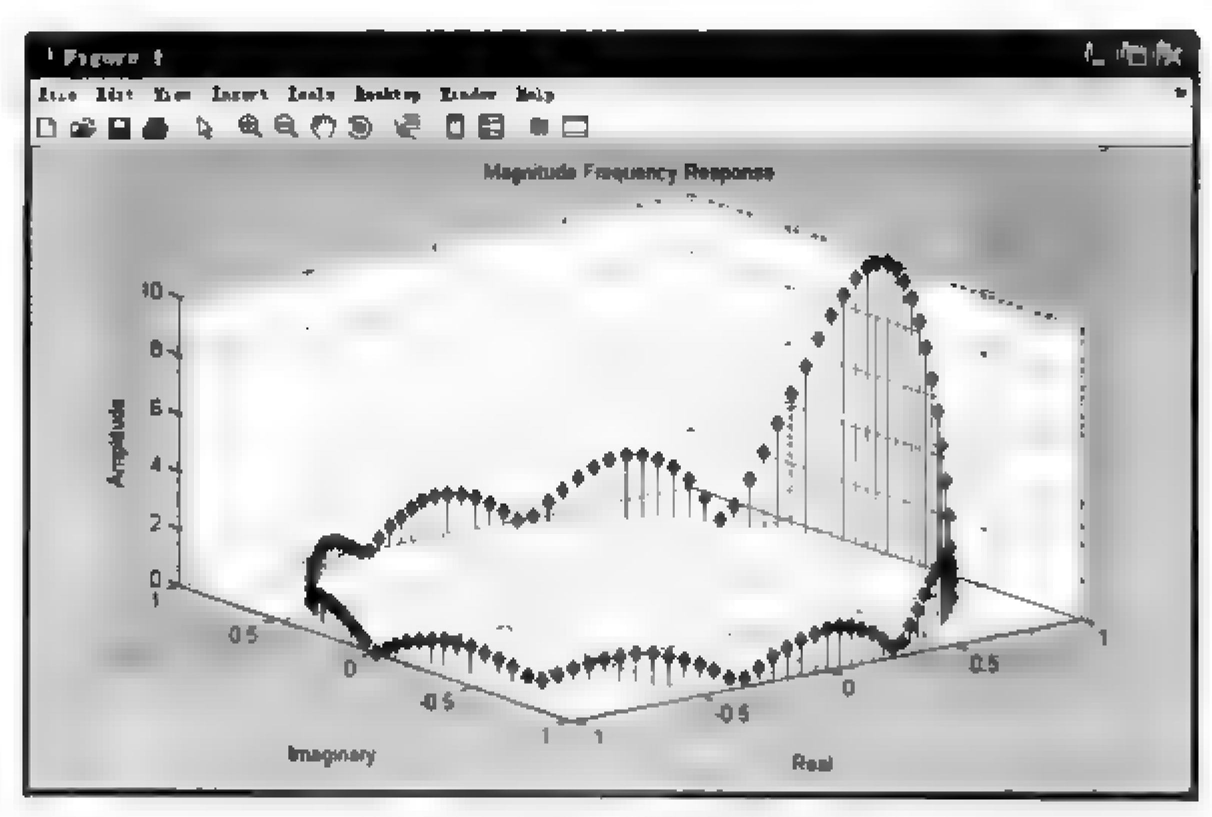


图 12-63 三维茎状图的绘制

例 12-26 使用 stairs 函数绘制梯形图。

解：在命令窗口输入如下程序，并按 Enter 键确认。

首先定义一个时间变化的函数，如下所示。

```
>> alpha = 0.01;
>>beta = 0.5;
>>t = 0:10;
>>f = exp(-alpha*t).*sin(beta*t);
>>
```

使用 stairs 函数来用梯形图显示该函数的数据，如下所示。

```
>>stairs(t,f)
>>hold on
>>plot(t,f,'--*')
>>hold off
>>
```

再对图形加以注释并设置坐标轴的极限。

```
>>label = 'Stairstep plot of e^{-(\alpha*t)} sin\beta*t';
>>text(0.5,-0.2,label,'FontSize',14)
>>xlabel('t = 0:10','FontSize',14)
>>axis([0 10 -1.2 1.2])
>>
```

运行结果如图 12-64 所示。

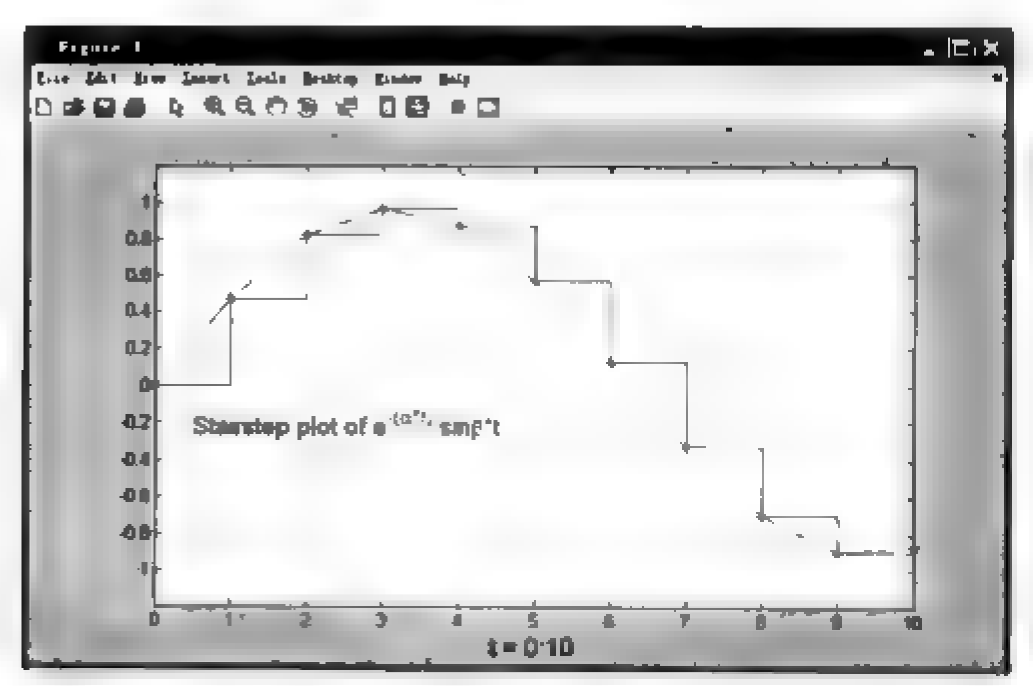


图 12-64　梯形图的绘制

4. 方向和速度矢量图形

MATLAB 7.0 提供了一些函数用于绘制方向矢量和速度矢量图形，这些函数有 compass、feather、quiver 和 quiver3。下面对这些函数的使用做详细介绍，如表 12-6 所示。

表 12-6　方向和速度矢量函数

函　　数	功能描述
compass	显示极坐标图形中的极点发散出来的矢量图
feather	显示从一条水平线上均匀间隔的点所发散出来的矢量图
quiver	显示由(u,v)矢量特定的二维矢量图
quiver3	显示由(u,v,w)矢量特定的三维矢量图

有关这些函数的具体使用方法，用户可以参照 MATLAB 7.0 的帮助系统，下面举例对它们的用法予以说明。

例 12-27　本例绘制一个 compass 图形，使用两个向量分别显示 12 小时内的风向和强度。

解：在命令窗口输入如下程序，并按 Enter 键确认。

```
>> wdir = [45 90 90 45 360 335 360 270 335 270 335 335];
>>knots = [6 6 8 6 3 9 6 8 9 10 14 12];
>> rdir = wdir * pi/180;
>> [x,y] = pol2cart(rdir,knots);
>>compass(x,y)
>>hold on
>> desc = {'Wind Direction and Strength at',
           'Logan Airport for ',
           'Nov. 3 at 1800 through',
           'Nov  4 at 0600'};
>>text(-28,15,desc)
>>
```

运行结果如图12-65所示。

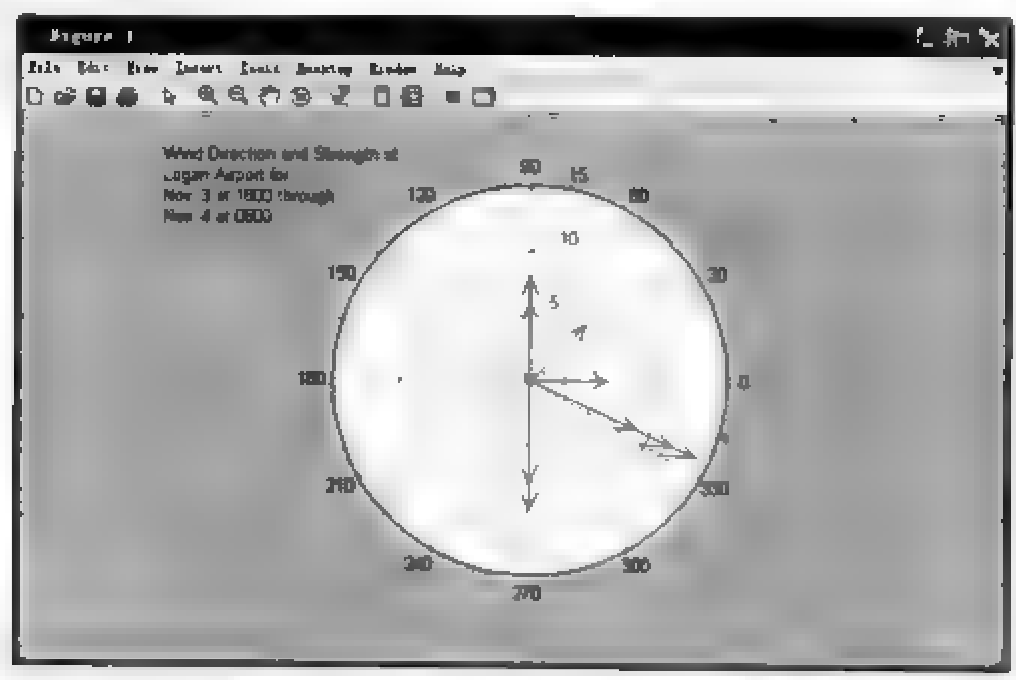


图12-65 compass图形的绘制

例12-28 使用quiver函数绘制箭头图形。

解：在命令窗口输入如下程序，并按Enter键确认输入，运行结果如图12-66所示。

```
>> % 该程序用于绘制箭头图形
>> [x,y] = meshgrid(-2:.2:2,-1:.15:1);
>> z = x .* exp(-x.^2 - y.^2);
>> [px,py] = gradient(z,.2,.15);
>> contour(x,y,z)
>>hold on
>> quiver(x,y,px,py)
>>hold off
>>axis image
>>
```

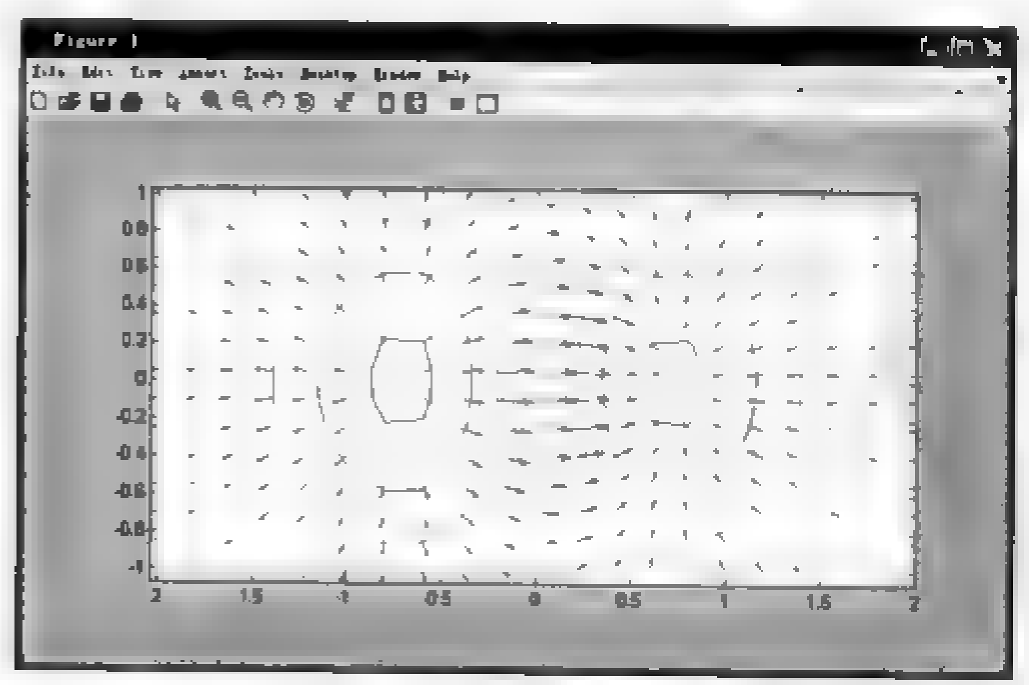


图12-66 quiver函数的使用

5. 等高线的绘制(Contour Plots)

等高线函数为创建、显示并标注由一个或多个矩阵确定的等值线，MATLAB 7.0提供

了如下函数来绘制等高线图形，如表 12-7 所示。

表 12-7　等高线绘制函数及其功能

函 数 名	功 能 描 述
clabel	使用等值矩阵生成标注，并将标注显示在当前图形
contour	显示矩阵 Z 的二维等高线图
contour3	显示矩阵 Z 的三维等高线图
contourf	显示矩阵 Z 的二维等高线图，并在各等高线之间用实体颜色填充
contourc	用于计算由其他等高线函数调用的等值矩阵
meshc	创建一个与二维等高线图匹配的网线图
surfc	创建一个与二维等高线图匹配的曲面图

有关各个函数的具体使用方法用户可以参见 MATLAB 7.0 的帮助系统，下面举例对其中的一些函数予以说明。

例 12-29　使用 contour 函数绘制等高线图形。

解：在命令窗口输入如下程序，并按 Enter 键确认输入，运行程序。

```
>> % 该程序用于绘制等高线图形
>> % 首先生成(x,y)坐标，然后生成 y 坐标，最后绘制等高线
>> [x,y]=meshgrid(1:100,1:100);
>> z=peaks(100);
>> contour(x,y,z)
>>
```

运行结果如图 12-67 所示。

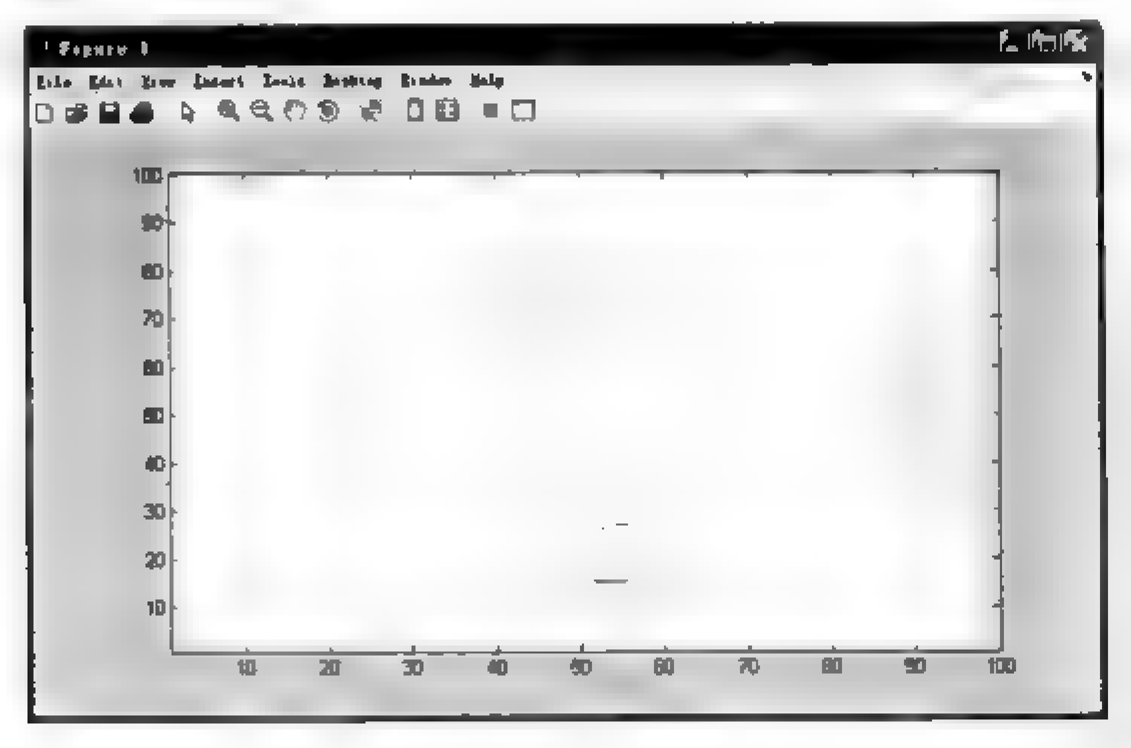


图 12-67　contour(x,y,z)命令的使用

例 12-30　使用 clabel 函数给等高线做标注。

解：在命令窗口输入如下程序，并按 Enter 键确认输入，运行程序。

```
>> Z = peaks;
>> [C,h] = contour(Z,10);
>> clabel(C,h)
>> title({'Contour Labeled Using','clabel(C,h)'})
>>
```

运行结果如图 12-68 所示。

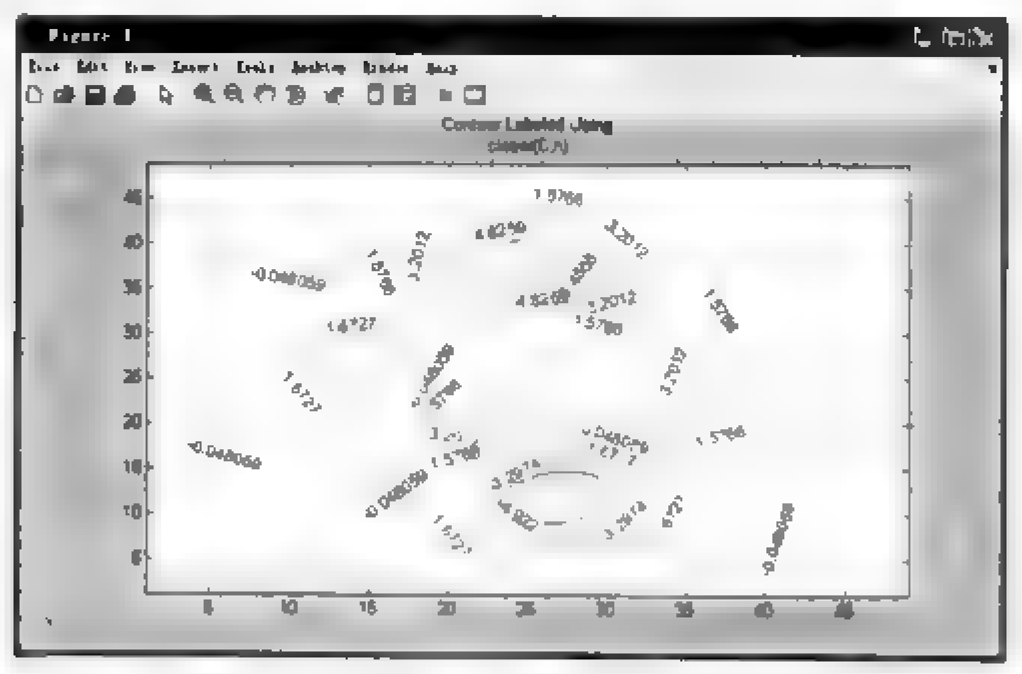


图 12-68 使用 clabel 函数给等高线做标注

例 12-31 使用 surfc 函数绘制具有等高线的三维表面图形。

解：在命令窗口输入如下程序，并按 Enter 键确认输入，运行程序。

```
>> % 该程序用于绘制具有等高线的三维表面表面图形
>> z=peaks(100);
>> surfc(z);
>>
```

运行结果如图 12-69 所示。

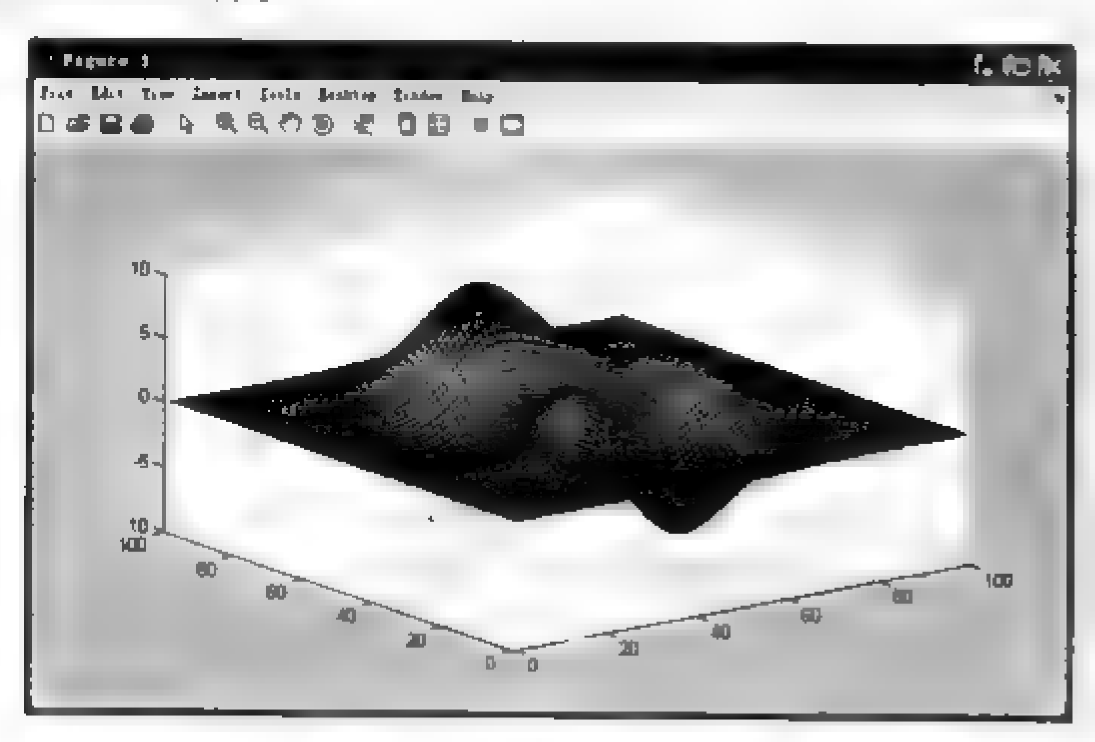


图 12-69 surfc 函数的使用

6. 网格图的绘制

MATLAB 7.0 提供了 mesh 函数绘制三维网格图形的绘制，它的使用格式如下。

- $mesh(x,y,z,c)$ 命令通过 4 个矩阵参数绘制彩色的三维网格图形。其中，图形的视口由 view 函数定义；图形的各轴范围由 x、y 和 z 或者通过当前的 axis 函数值定义；图形的颜色范围由 C 或者当前的 caxis 值定义；
- $mesh(x,y,z)$ 命令使用 $c=z$，因此图形的颜色随高度按比例变化；
- $mesh(x1,y1,z)$ 命令和 $mesh(x1,y1,z,c)$ 命令，使用两个向量 $x1$ 和 $y1$ 代替矩阵 x 和 y，必须满足条件 $length(x1)=n$ 和 $length(y1)=m$，而 $size(z)=[m,n]$。在这种情况下，网格线上的点由坐标(x(j), y(i), Z(i,j))决定。注意，向量 x 对应于矩阵 z 的列，向量 y 对应矩阵 z 的行；
- $mesh(z)$ 命令和 $mesh(z,c)$ 命令默认 $x=1:n$ 和 $x=1:m$。在这种情况下，高度 z 是一个单值函数；
- $mesh(...,'propertyname',propertyvalue)$ 命令设置图形表面的属性值，单个语句可以设定多个属性值。

下面举例对 mesh 函数的使用方法进行说明。

例 12-32　使用 mesh 函数绘制网格图。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>>[X,Y,Z]=peaks(30);
>> mesh(X,Y,Z)
>> grid,xlabel( 'x-axis'),ylabel( 'y-axis'),zlabel('z-axis')
>> title( 'MESH of PEAKS')
>>
```

运行结果如图 12-70 所示。

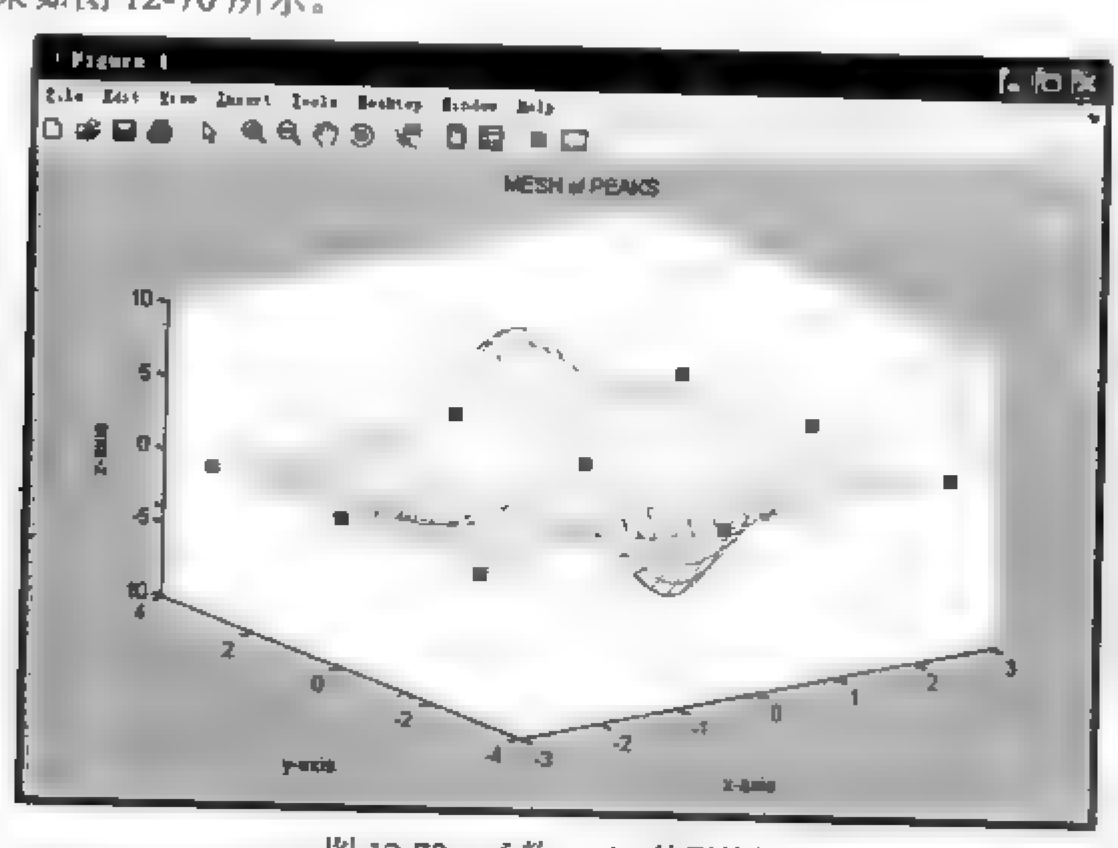


图 12-70　函数 peaks 的网格图

在显示器上要注意到线的颜色与网格的高度有关。一般情况下，函数 mesh 有可选的参量来控制绘图中所用的颜色。关于 MATLAB 7.0 如何使用、改变颜色在下一节讨论。在任何情况下，由于颜色用于增加图形有效的第四维，这样使用的颜色被称做伪彩色。

除了上例中的输入参量，函数 mesh 和大多数三维绘图函数都可按多种输入参量调用。这里所用的句法是最详细的，它给出了所有 3 个坐标轴的信息。最通常的变更方法是使用向量，将它传递给 meshgrid，生成 x 与 y 坐标轴，比如 mesh(x,y,z)。有关其他调用语法形式的信息，参阅 MATLAB 7.0 参考指南或在线帮助。

如上图所示，网格线条之间的区域是不透明的，命令 hidden 控制网格图的这个特性。

例 12-33　使用 MATLAB 7.0 的函数 hidden 网格线条之间的区域是否透明。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>>[X,Y,Z]=sphere(12);
>>subplot(1,2,1)
>>mesh(X,Y,Z),title( 'Opaque')
>>hidden on
>>axis off
>>subplot(1,2,2),title( 'Transparent')
>>mesh(X,Y,Z)
>>hidden off
>>axis off
>>
```

运行结果如图 12-71 所示。

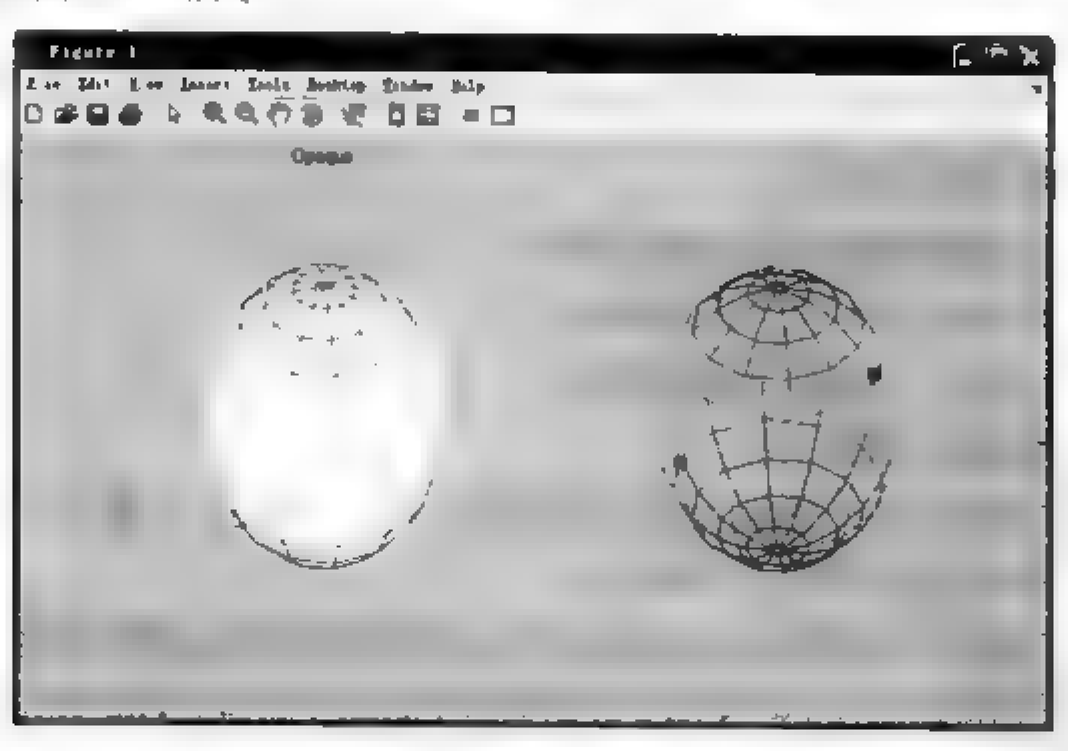


图 12-71　使用 hidden 函数控制网格线条之间的区域是否透明

此外，MATLAB 7.0 还提供了 meshc 函数绘制带有等高线的三维网格图形。meshz 函数绘制增加了边界面屏蔽的三维网格图。它们的调用形式与 mesh 函数相同。

例 12-34　使用 meshz 函数绘制具有等高线的三维网格图形。

解：在命令窗口输入如下程序，并按 Enter 键确认输入，运行结果如图 12-72 所示。

```
>> % 该程序用于绘制具有等高线的三维网格图形
>> z=peaks(50);
>> meshc(z)
>>
```

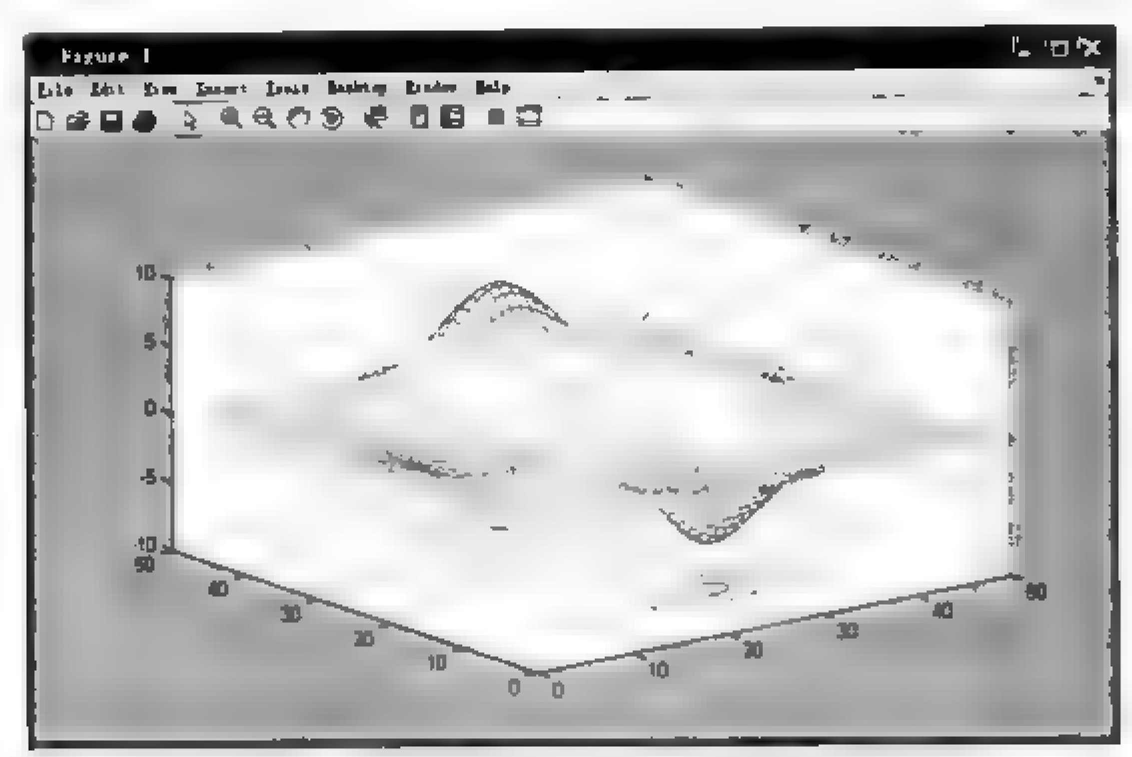


图 12-72　带等高线的三维网格图形

12.2　交互式绘图操作

前边已经介绍了一些常规的绘图方式，用户可以使用这些方法进行基本的图形绘制，并进行图形注释和特殊图形的绘制。此外，MATLAB 7.0 还提供了一种交互式的绘图方式，使用这种方式进行绘图和图形注释将更为方便。交互式绘图方式可以实现如下操作。

- 创建各种类型的图形；
- 从工作区间选择变量来直接进行绘制；
- 在图形中创建并多子图并进行操作；
- 添加诸如箭头、直线或文本等注释；
- 对图形对象进行属性设置。

单击图形工具栏上的按钮或是在命令窗口中输入 plottools 命令，都将在图形窗口周围打开各种图形工具，如图 12-73 所示。

单击图形工具栏上的按钮，MATLAB 7.0 将关闭图形窗口周围的各种图形工具。

同时，用户也可以从 View 菜单中分别调出 Figure Palette、Plot Browser 和 Property Editor 等 3 个板块。

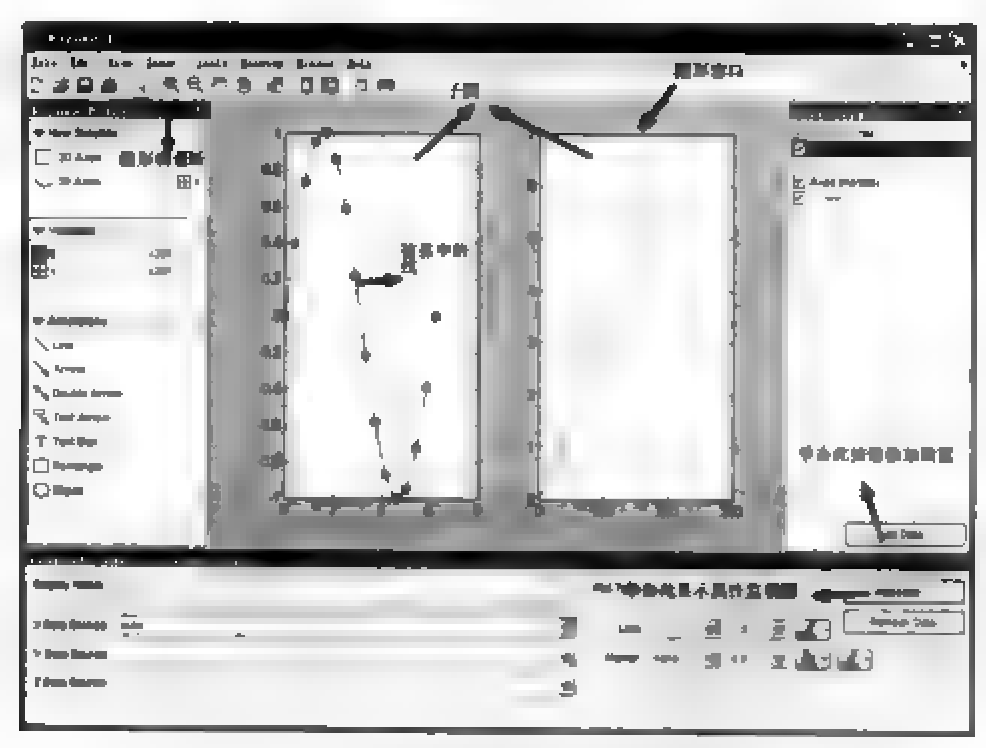


图 12-73　图形窗口周围的各种图形工具

下面就对这 3 个板块分别予以介绍。

1. Figure Palette 板块

Figure Palette 包括 3 个子板块，用户可以单击相应的按钮来选择所需的板块，选中之后的板块将展开其内容。

Figure Palette 可以让用户使用响应的板块进行如下操作。

- New Subplots – 给图形中增添二维或三维图形。
- Variables – 浏览工作区间的变量并绘制图形。
- Annotations – 给图形增添注释。

(1) 增添多子图

用户可以使用 New Subplots 板块创建多子图，用户只需单击图形类型后边的网格图标，MATLAB 7.0 将拉出一个网格，随着鼠标的移动，部分网格将变成灰色，这些灰色的网格就是用户即将创建的多子图布局。而单击网格下边的 Cancel 可以使图形保持初始状态，如图 12-74 所示。

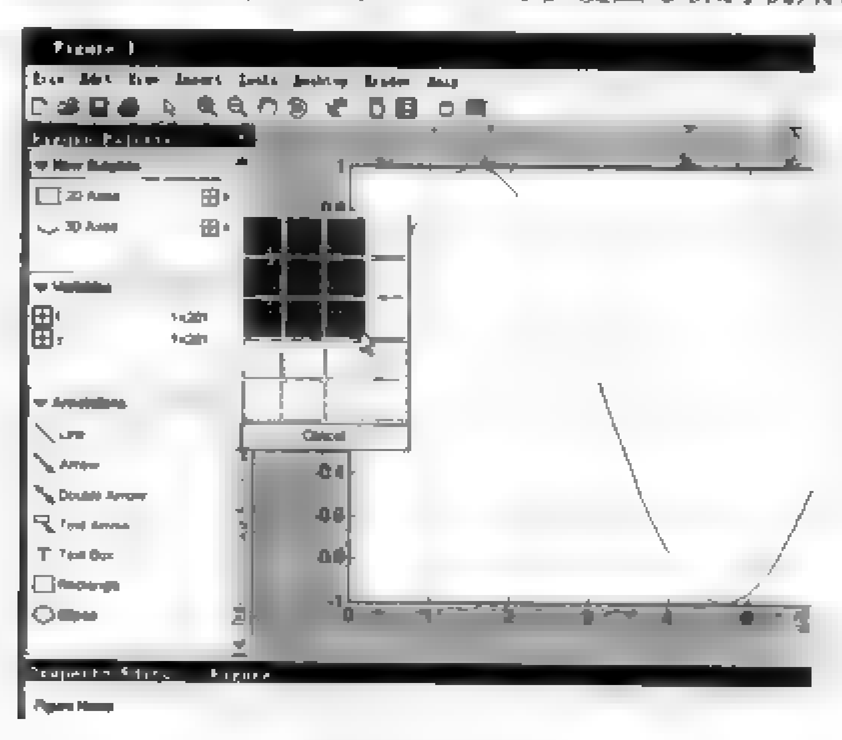


图 12-74　增添多子图

此图所示为 9 个大小相等的子图，初始的图形已经自动调整了尺寸来适应新的图形布局。

(2) 绘制工作区间的变量

Variables panel 显示的是当前工作区间的变量，双击该板块中的一个变量，将在激活的图形中绘制该变量的曲线。如果右击其中的一个变量，将调出关联菜单，用户可以选择需要绘制的图形种类。

例如，图 12-75 所示为如何绘制矩阵 Z 的条形图，这相当于将矩阵传递给绘图函数。

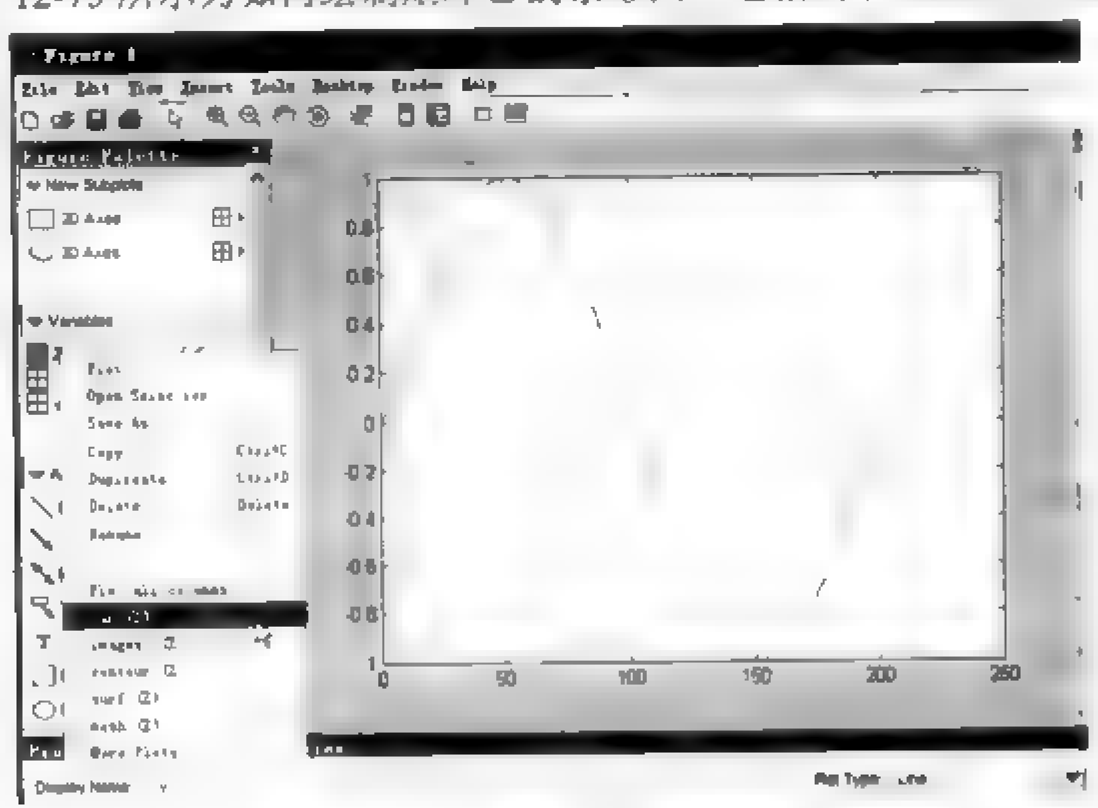


图 12-75　绘制工作区间的变量

2. Plot Browser 板块

Plot Browser 板块给图形窗口中的每一个图形都提供了图例，它列出了所有图形以及绘制图形的对象(线型、颜色等)，如图 12-76 所示。

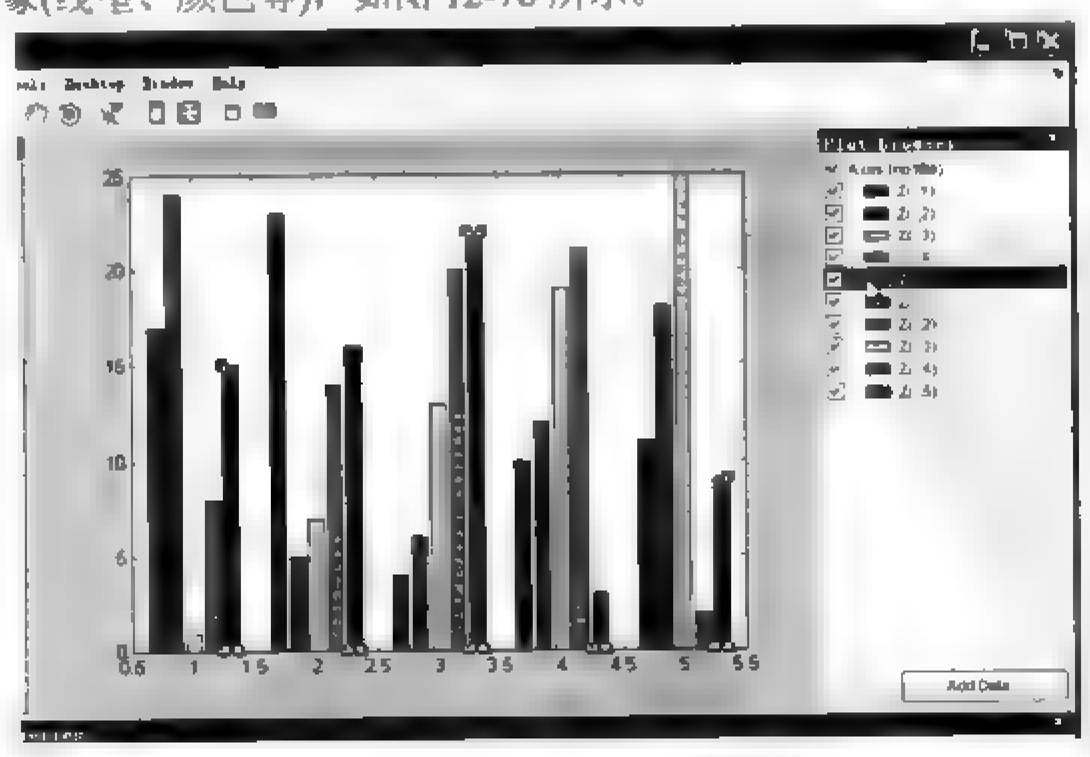


图 12-76　Plot Browser 板块示意图

Plot Browser 板块可以对图形进行简单的编辑，也可以给图形增添新的数据，下面分

步予以说明。

(1) 编辑功能

如果用户想修改单个条形图的属性，可以双击 Plot Browser 板块图中的颜色框，此时对应条形图的属性将在 Property Editor 板块中予以显示，Property Editor 板块位于图形窗口的下方。

用户如果在图形窗口中选中一个条形图，那么 Plot Browser 板块中对应的图例将处于亮显状态；同理，选中 Plot Browser 板块中的某个图例，图形窗口中对应的条形图也将处于选中状态。

此外，Plot Browser 板块中的复选框可以用来控制对象的可见性，当复选框处于选中状态时，MATLAB 7.0 被绘制出对应的对象，否则该对象将被隐藏。而当用户想删除某个对象时，只需右击该对象，在弹出的菜单中选择 Delete 项即可。

(2) 增添数据

Plot Browser 板块也提供了给图形添加数据的机制，其使用程序如下。

- 从 New Subplots 子板块中选定需要增添的二维图形或三维图形的数目。
- 创建了图形之后，在 Plot Browser 板块中选中它，使得板块底部的 Add Data 按钮处于激活状态，单击 Add Data 按钮将调出 Add Data to Axes 对话框，如图 12-77 所示。
- 用户可以利用 Add Data to Axes 对话框来选择图形样式，并从工作区间中选择变量传递给绘图函数。用户也可以自己指定一个 MATLAB 7.0 数学表达式，用来给所要绘制图形的数据进行定义。

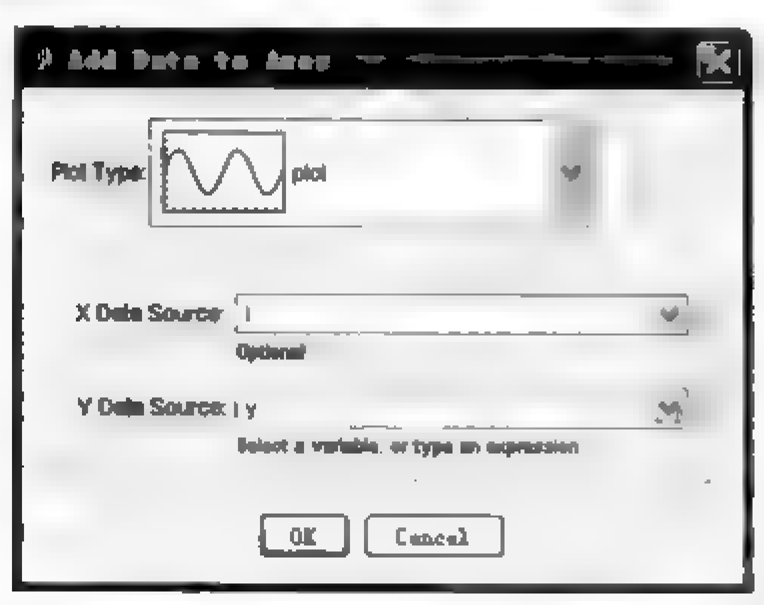


图 12-77　Add Data to Axes 对话框

3. Property Editor 板块

用户可以利用 Properties Editor 板块来定义所选对象的属性，当没有对象被选定时，Property Editor 板块将显示图形的属性。

可以使用 4 种方法调出 Property Editor 板块。

- 当图形编辑模式处于激活状态时，双击所以进行编辑的对象。
- 选中某个物体并右击，从弹出的快捷菜单中选择 Properties 选项。
- 从 View 菜单中选择 Property Editor 选项。

◆ 直接在命令窗口中输入 propertyeditor 命令。

调出的 Property Editor 板块如图 12-78 所示。

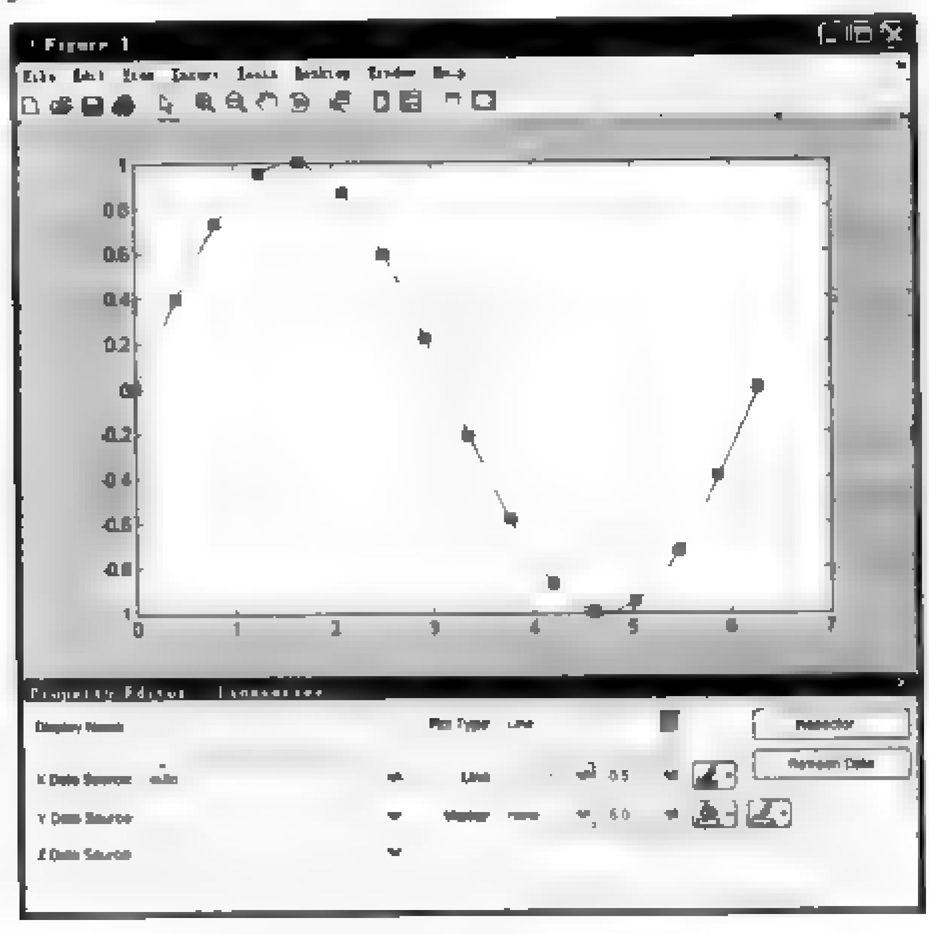


图 12-78　Property Editor 板块

用户可以使用该板块对图形及其对象进行各种图例注释，并切换绘图的类型。在此不再赘述。

12.3　图形的高级控制

前面已经介绍了对图形的一些基本操作，并对图形窗口有了基本的了解，掌握这些操作之后，用户可以对图形进行简单的编辑。但是，MATLAB 7.0 对图形的处理远远不止上述简单操作，它给我们提供了很多编辑图形的高级的工具，掌握这些工具，将给我们对图形的处理操作带来很大的便利。本节将具体介绍 MATLAB 7.0 图形的高级控制及其使用方法。

12.3.1　视点控制和图形的旋转

为了使图形的效果更逼真，有时需要从不同的角度观看图形，MATLAB 7.0 语言提供了 view、viewmtx 和 rotate3d 等 3 个命令进行操作。用户可以在命令窗口中调用这 3 个函数。其中，view 函数主要是从不同的角度观察图形；viewmtx 给出指定视角的正交转换矩阵；而 rotate3d 函数可以让用户方便地用鼠标来适时旋转视图。

下面具体介绍一下 view 函数的使用方法，如下所示。

- $view(az,el)$ 命令和 $view([az,el])$ 命令设置了观看三维图的 3 个角度。其中 az 是水平方位角，从 Y 轴负方向开始，以逆时针方向旋转为正； el 是垂直方位角，向 Z 轴方向的旋转为正，向 Z 轴负方向的旋转为负。
- 默认的三维图的视角为：$az=-37.5$，$el=30$；
- 而当 $az=0$，$el=90$ 时，即俯视图形，其效果观看二维图形；
- $view([x,y,z])$ 命令设置在笛卡尔坐标系下的视角，而忽略向量 x、y 和 z 的幅值；
- $view(2)$ 命令也可以设置二维图形视角，即 $az=0$，$el=90$；
- $view(3)$ 命令设置三维图形视角，即 $az=-37.5$，$el=30$；
- $[az,el]=view$ 命令返回当前的 az 和 el 值。

关于 viewmtx 和 rotate3d 函数的使用方法用户可以参见 MATLAB 7.0 的帮助系统，在此不再赘述。

例 12-35　view 函数的使用。

解：该程序显示在不同的视角下观察三维图形的效果，在命令窗口输入如下程序，并按 Enter 键确认输入。

```
>> [X,Y]=meshgrid([-4:0.2:4]);
>> Z=exp(-0.5*(X.^2+Y.^2));
>> surf(X,Y,Z)
>>
```

运行结果如图 12-79 所示。

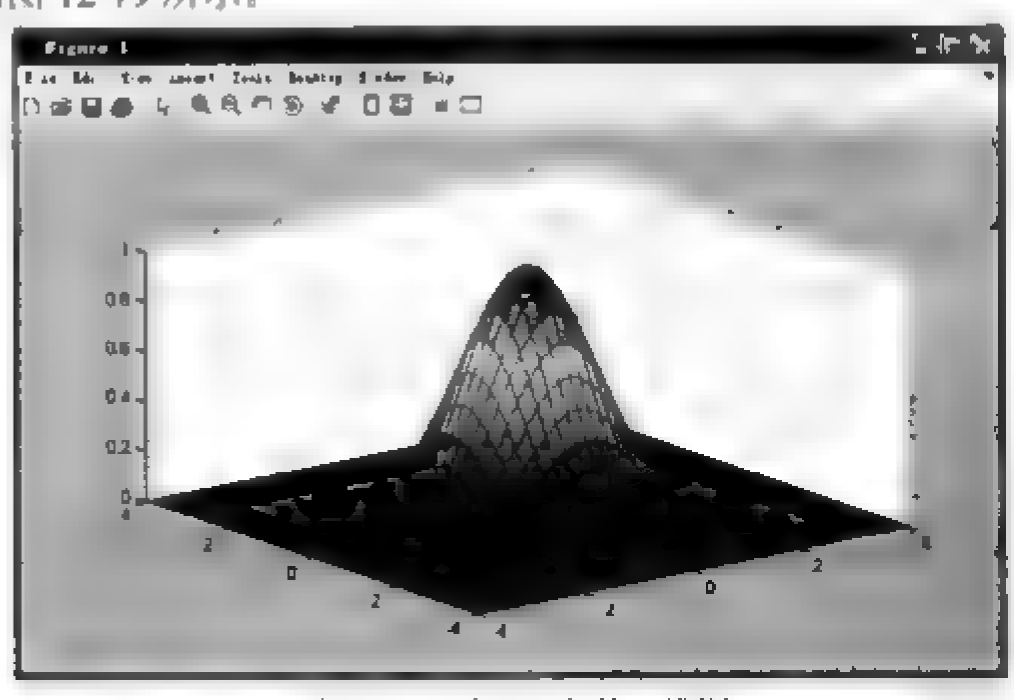


图 12-79　默认视角的三维图

继续在命令窗口输入如下程序，并按 Enter 键确认输入，运行结果如图 12-80 所示。

```
>> view(2)
>>
```

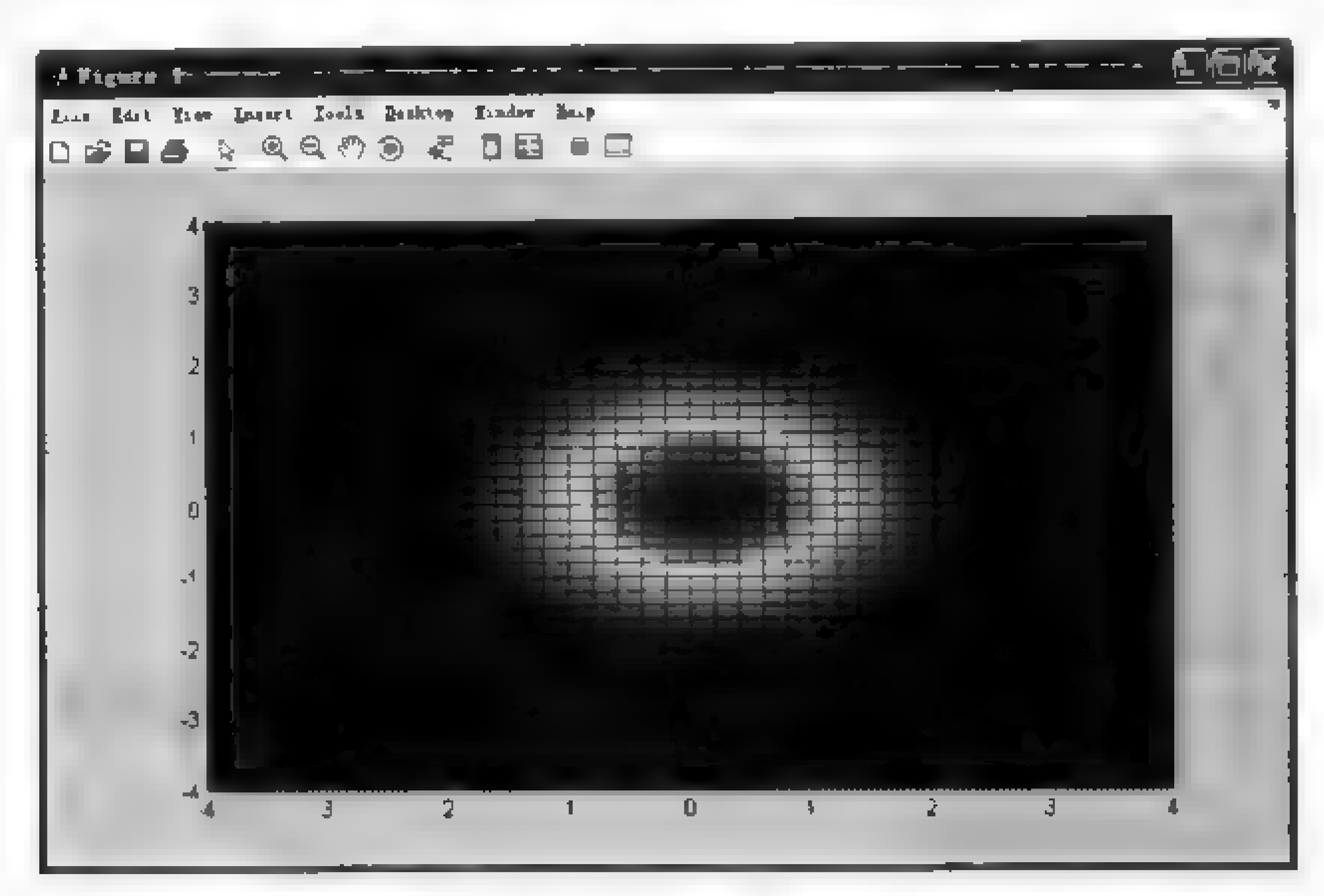


图 12-80　三维图的二维视角

继续在命令窗口输入如下程序，并按 Enter 键确认输入，运行结果如下所示。

```
>> [AZ,EL] = VIEW
AZ =
     0
EL =
    90
>>
```

可见，*[az,el] = view* 命令返回当前的 *az* 和 *el* 值。

关于 viewmtx 和 rotate3d 函数的使用方法用户可以参见 MATLAB 7.0 的帮助系统，在此不再赘述。

12.3.2　颜色的使用

本节从介绍颜色映像开始，讲述如何使用、显示、修改和创建用户自己的颜色映像。

1. 颜色映像理解

MATLAB 7.0 有一个叫颜色映像的数据结构来代表颜色值。颜色映像定义为一个有 3 列和若干行的矩阵。利用 0 到 1 之间的数，矩阵的每一行都代表了一种色彩。任一行的数字都指定了一个 RGB 值，即红、黄、蓝 3 种颜色的强度，形成一种特定的颜色。一些有代表性的 RGB 值如表 12-8 所示。

表 12-8　MATLAB 7.0 的映像元素

Red(红)	Green(绿)	Blue(蓝)	颜　色
0	0	0	黑
1	1	1	白
1	0	0	红
0	1	0	绿
0	0	1	蓝
1	1	0	黄
1	0	1	洋红
0	1	1	青蓝
2/3	0	1	天蓝
1	1/2	0	橘黄
.5	0	0	深红
.5	.5	.5	灰色

此外，还有 10 个 MATLAB 7.0 函数产生预定的颜色映像，如表 12-9 所示。

表 12-9　MATLAB 7.0 的常用映像元素

函　数	功 能 描 述
hsv	色彩饱和值(以红色开始和结束)
hot	从黑到红到黄到白
cool	青蓝和洋红的色度
pink	粉红的彩色度
gray	线性灰度
bone	带一点蓝色的灰度
jet	hsv 的一种变形(以蓝色开始和结束)
copper	线性铜色度
prim	三棱镜，交替为红色、橘黄色、黄色、绿色和天蓝色
flag	交替为红色、白色、蓝色和黑色

按照默认设置，上面所列的各个颜色映像产生一个 64×3 的矩阵，指定了 64 种颜色 RGB 的描述。这些函数都接受一个参量来指定所产生矩阵的行数。比如 hot(m)产生一个 m×3 的矩阵，它包含的 RGB 颜色值的范围从黑经过红、橘红和黄，到白。

大多数计算机在一个 8 位的硬件查色表中一次可以显示 256 种颜色，当然有些计算机的显示卡可以同时显示更多的颜色。这就意味着在不同的图中，一般一次可以用 3 个或 4 个 64×3 的颜色映像。如果使用了更多的颜色映像输入项，计算机必须经常在它的硬件查色表中调出输入项。比如，当在绘制 MATLAB 7.0 图形时背景图案发生了变化，就是发生了这种情况。所以，除非计算机有一次显示更多种颜色的显示卡，最好任何一次所用的颜色映像输入项数都小于 256。

2. 颜色映像使用

语句 colormap(M)将矩阵 M 作为当前图形窗口所用的颜色映像。例如，colormap(cool)装入了一个有 64 个输入项的 cool 颜色映像。colormap　default 装入了默认的颜色映像(hsv)。

函数 plot、plot3、contour 和 contour3 不使用颜色映像，它们使用列在 plot 颜色和线形表中的颜色。而大多数其他绘图函数，比如 mesh、surf、fill、pcolor 和它们的各种变形函数，使用当前的颜色映像。

接受颜色参量的绘图函数中的颜色参量通常采用以下 3 种形式之一。

(1) 字符串，代表 plot 颜色或线型表中的一种颜色，比如 'r' 代表红色。

(2) 3 个输入的行向量，它代表一个单独的 RGB 值，比如[.25　.50　.75]。

(3) 矩阵，如果颜色参量是一个矩阵，其元素作了调整，并把它们用作当前颜色映像的下标。

3. 颜色映像显示

可以用多种途径来显示一个颜色映像。其中一个方法是观察颜色映像矩阵的元素。例如，在命令窗口中输入如下数据，并按 Enter 键确认，得到如下结果。

```
 >> hot(8)
ans =
    0.3333         0         0
    0.6667         0         0
    1.0000         0         0
    1.0000    0.3333         0
    1.0000    0.6667         0
    1.0000    1.0000         0
    1.0000    1.0000    0.5000
    1.0000    1.0000    1.0000
>>
```

上面的数据显示出第一行是 1/3 红色，而最后一行是白色。

此外，函数 pcolor 可以用来显示一个颜色映像。例如，在命令窗口中输入如下数据，并按 Enter 键确认。

```
>>n=16;
>> colormap(jet(n))
>> pcolor([1:n+1;1:n+1]')
>> title( 'Using Pcolor to Display a Color Map' )
>>
```

运行结果如图 12-81 所示。

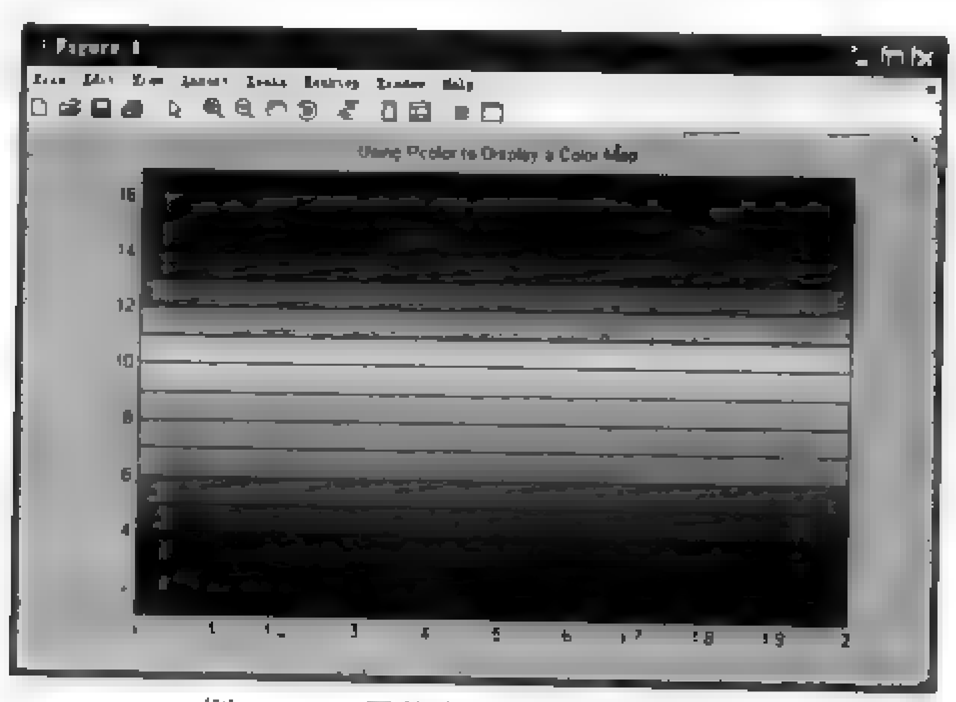


图 12-81 用伪彩色来显示颜色映像

最后，函数 colorbar 在当前的图形窗口中增加水平或垂直的颜色标尺以显示当前坐标轴的颜色映像。

- *colorbar('horiz')* 在当前的图形下面放一个水平的颜色条。
- *colorbar('vert')* 在当前的图形右边放一个垂直的颜色条。
- 对无参量的 *colorbar* 命令，如果当前没有颜色条就加一个垂直的颜色条，或者更新现有的颜色条。

下面的例子就演示了 colorbar 的用法。

```
>> [x,y,z]=peaks;
>>mesh(x,y,z);
>>colormap(hsv)
>> axis([-3  3  -3  3  -6  8])
>> colorbar
>>
```

运行结果如图 12-82 所示。

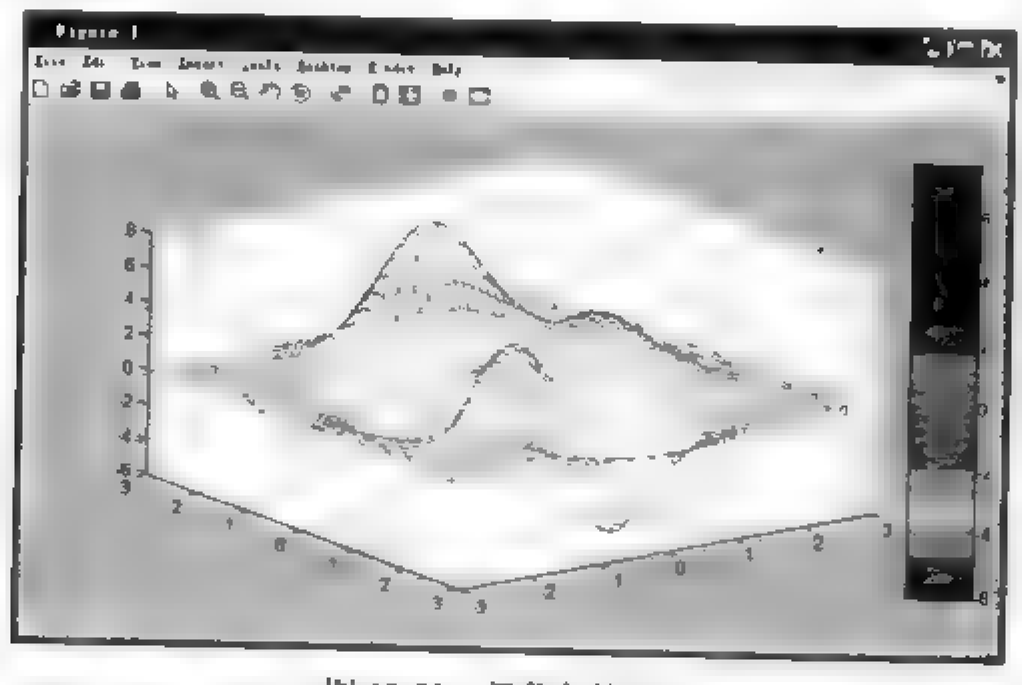


图 12-82 颜色条的使用

4. 颜色映像的建立和修改

颜色映像就是矩阵，意味着你可以象其他数组那样对它们进行操作。函数 brighten 就利用这一点通过调整一个给定的颜色映像来增加或减少暗色的强度。它的使用格式如下。

- $brighten(n)(0 < n <= 1)$ 命令使当前颜色映像变亮。
- $brighten(n)(-1 <= n < 0)$ 命令使当前颜色映像变暗。
- $brighten(n)$ 后加一个 $brighten(-n)$ 使颜色映像恢复原来状态。
- $newmap = brighten(n)$ 命令创建一个比当前颜色映像更暗或者更亮的新的颜色映像，而并不改变当前的颜色映像。
- $newmap = brighten(cmap, n)$ 对指定的颜色映像创建一个已调整过的式样，而不影响当前的颜色映像或指定的颜色映像 $cmap$ 。

用户可以通过生成 m×3 的矩阵 mamap 来建立自己的颜色映像，并用 colormap(mymap) 来安装它。颜色映像矩阵的每一个值都必须在 0 和 1 之间。如果企图用大于或小于 3 列的矩阵或者包含着比 0 小比 1 大的任意值，函数 colormap 会提示一个错误然后退出。

用户也可以在算术上来组合颜色映像，虽然结果有时是不可预料的。比如，一个叫 pink 的颜色映像仅仅是：

```
>>pinkmap=squr(2/3*gray+1/3*hot);
```

只当所有元素都在0与1之间时，才能保证结果是一个有效的颜色映像。精通 MATLAB 7.0 工具箱中包含了一个名叫 rainbow 的颜色映像，它把可视范围扩展到整个颜色映像。精通 MATLAB 7.0 工具箱中还包含了一个名叫 mmap 的函数，它可以根据用户所提供的颜色建立一个单色(比如粉红、灰色或铜黄色)的颜色映像。关于这两个函数的具体使用方法用户可以参见 MATLAB 7.0 的帮助系统。

12.3.3 光照控制

MATLAB 7.0 语言提供了许多函数在图形中进行对光源的定位并改变光照对象的特征，如表 12-10 所示。

表 12-10　MATLAB 7.0 语言的图形光源操作函数

函 数 名	功 能 描 述
camlight	设置并移动关于摄像头的光源
lightangle	在球坐标下设置或定位一个光源
light	设置光源
lighting	选择光源模式
material	设置图形表面对光照的反映模式

其中，light 函数用于设置光源，它的调用格式如下。

bright(*param*1,*value*1,..., *paramN*, *valueN*) 命令实现对光源参数的设置，当没有参数输入时，将以默认值进行设置。光源设置有 3 个很重要的参数，如下所示。

- Color：光源对象投射的光的颜色。
- Style：点光源或是平行光。
- Position ：点光源的位置或是平行光的方向。

例 12-36　本例首先绘制一个膜面图，然后使用位置向量 [0 -2 1] 设置光源的方向。

解：在 MATLAB 命令窗口中输入如下命令，并按 Enter 键确认。

```
>> membrane
>>
```

生成的膜面图如图 12-83 所示。

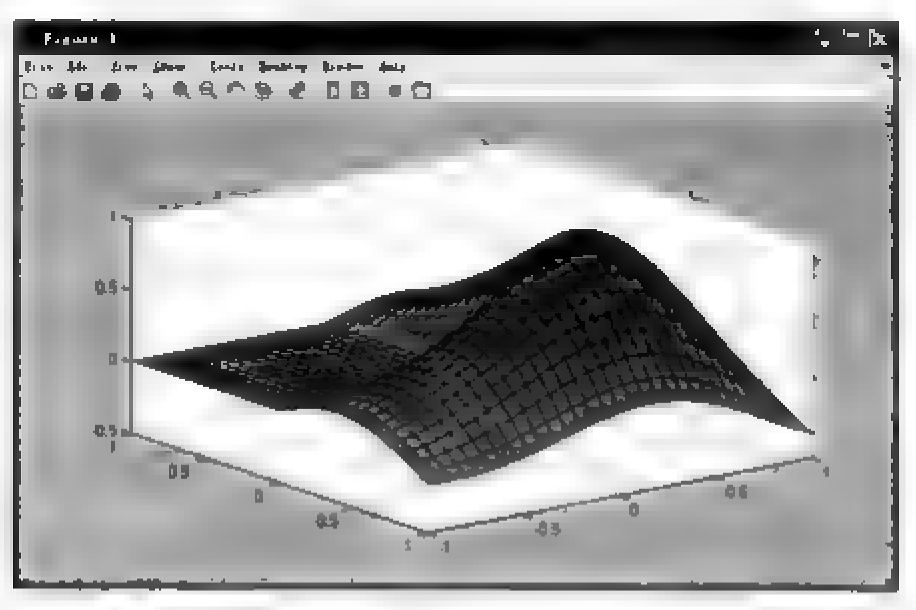


图 12-83　普通的膜面图

现在给膜面增加光源，在命令窗口中继续输入如下命令，并按 Enter 键确认。

```
>> light('Position',[0 -2 1])
>>
```

运行结果如图 12-84 所示。

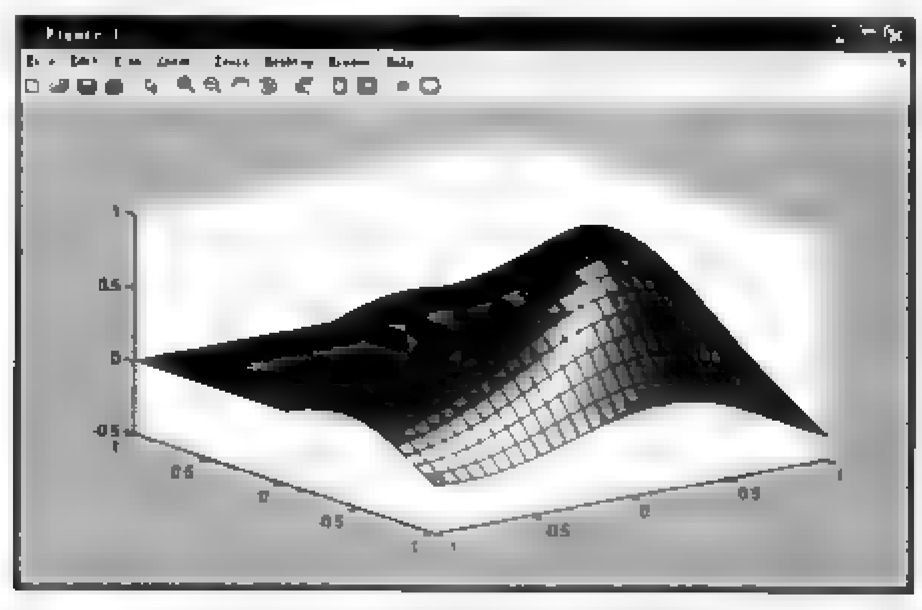


图 12-84　增加光源之后的膜面图

12.4 习　题

1. 熟悉图形窗口的各个菜单和工具栏的功能和作用。

2. 编制 MATLAB 7.0 程序，该程序绘制两条曲线，x 的取值在[0,2pi]，以 π/10 为步长，一条是正弦曲线，一条是余弦曲线，线宽为 6 个象素，正弦曲线为绿色，余弦曲线为红色。线型分别为实线和虚线。

3. 使用交互式绘图方式对上题所绘的图形进行修改，将数据点型换为圆圈。

4. 使用 MATLAB 命令语句给习题 12.2 所绘制的图形增添 x 轴标签，使用交互式绘图方式给习题 12.2 所绘制的图形增添 y 轴标签。

5. 分别使用命令语句和交互式绘图方式给上题的图形增添图题，图题为“一个波长的正弦曲线和余弦曲线”。

6. 给上题所绘的两条曲线增添图例，分别为“正弦曲线”和“余弦曲线”。

7. 使用箭头、矩形和文本框对上题所绘制的图形进行标注，用矩形圈出图形在 x=π 处的点，并用箭头指向文本框，文本框中表示该点处的 x 和 y 的值。

8. 重新绘制习题 12.2 所描述的曲线，将正弦曲线和余弦曲线分别画在两个子图中，子图竖向排列。

9. 编制一个程序，该程序将接收输入变量 c，c 为一个复数，然后在卡式坐标系中绘制一条从原点到 c 的直线，直线采用星形点型，虚线绘制。

10. 绘制函数 $v(t)=10e^{(-0.2+\pi)t}$，其中 $0\le t\le 10$，使用函数 $plot(t,v)$ 绘制。

11. 仍然绘制习题 12.10 所示曲线，使用函数 $plot(v)$ 进行绘制，并与上例进行比较。

12. 在极坐标下绘制函数 $v(t)=10e^{(-0.2+\pi)t}$，其中 $0\le t\le 10$。

13. 绘制三维曲线 $z=e^{2x+3y}$。$0\le x\le 10$，$y=\sin(x)$。

14. 绘制函数 $z=e^{x+y^i}$ 的网格图、面图和等高线图，其中 $-1\le x\le 1$。

15. 将习题 12.13 和习题 12.14 的图形分别逆时针旋转 45 度。

16. 利用表 12-10 所示的函数，改变上题中图形的颜色映像。

第13章 句 柄 图 形

句柄图形是对底层图形例程集合的总称，它实际上是进行生成图形的工作。这些细节通常隐藏在图形 M 文件的内部，但如果想使用它们也是可得到的。

句柄图形允许用户定制图形的许多特性，而这用高级命令和前几章里描述的函数是无法实现的。例如，如果想用橘黄色来画一条线，而不是 plot 命令中可用的任何一种颜色，就可以用句柄图形提供的一种方法。

本章将介绍句柄图形的基本概念，让初学者也可以利用句柄图形。在本章最后给出了关于句柄图形对象属性和它们的值，这很有实用意义。

13.1 句柄图形对象

句柄图形是基于这样的概念，即一幅图的每一组成部分是一个对象，每一个对象有一系列句柄和它相关，每一个对象又按需要可以改变的属性。

面向对象的编程语言，数据库对象，操作系统和应用程序接口都用到了对象的概念。一个对象可以被粗略地定义为由一组紧密相关、形成惟一整体的数据结构或函数集合。在 MATLAB 7.0 中，图形对象是一幅图中很独特的成分，它可以被单独地操作。

由图形命令产生的每一件东西都是图形对象。它们包括图形窗口或仅仅说是图形，还有坐标轴、线条、曲面、文本和其他。这些对象按父对象和子对象组成层次结构。计算机屏幕是根对象，并且是所有其他对象的父亲。图形窗口是根对象的子对象；坐标轴和用户界面对象(在下一章讨论)是图形窗口的子对象；线条、文本、曲面、补片和图象对象是坐标轴对象的子对象。这种层次关系在如图 13-1 所示。

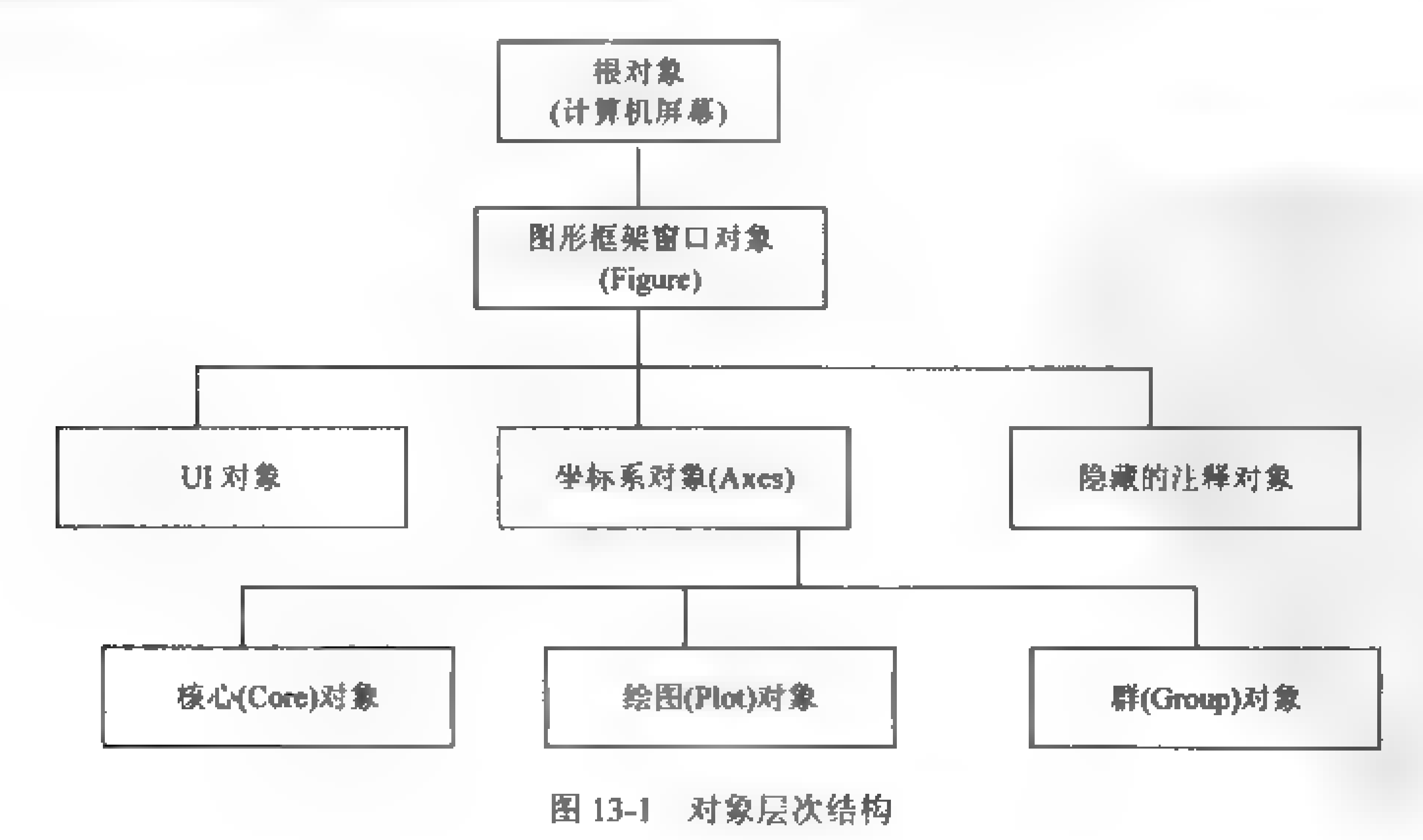


图 13-1 对象层次结构

根可包含一个或多个图形窗口，每一个图形窗口可包含一组或多组坐标轴。所有其他的对象(除了在下一章讨论的UI对象除外)都是坐标轴的子对象，并且在这些坐标轴上显示。所有创建对象的函数当父对象或对象不存在时，都会创建它们。例如，如果没有图形窗口，*plot*(*rand*(*size*([1：10])))函数会用默认属性创建一个新的图形窗口和一组坐标轴，然后在这组坐标轴内画线。

下面就对这些句柄图形对象做一些简单的介绍。

13.1.1　图形框架窗口对象(Figure)

图形框架窗口对象是 MATLAB 7.0 显示图形的窗口，包括菜单、工具栏、交互式对象、弹出式菜单、坐标、坐标轴子对象以及其他的图形对象。

MATLAB 7.0 对一次打开的图形数目没有限制(用户的计算机系统可能会做出限制)。

在 MATLAB 7.0 中，图形框架窗口对象有两条特殊的作用。

- 包含数据图形；
- 包含图形用户界面操作 GUI。

图形框架窗口可以同时包含图形和 GUI 组件，而该 GUI 可能是设计成绘制数据的，因此可能会同时包括坐标轴和交互式对象。

1．绘制图形的图形框架窗口对象(Figures Used for Graphing Data)

在调用 MATLAB 7.0 的绘图函数(如 plot 和 surf 等)时，如果此前没有图形存在，系统将自动创建一个图形，如果同时有多个图形打开，其中必有一个图形处于当前状态，该图形就是图形输出的目标。

实现对图形对象句柄的访问是实现句柄图形操作的前提，表 13-1 列出了 MATLAB 7.0 语言中实现句柄访问的函数。

表 13-1　句柄访问函数及其功能

函 数 名	功 能 描 述
gca	获得当前坐标轴对象的句柄
gcbf	获得当前正在执行调用的图形对象的句柄
gcbo	获得当前正在执行调用的对象的句柄
gcf	获得当前图形对象的句柄
gco	获得当前对象的句柄

那么，由上表可以得知，要获得当前图形框架窗口的对象，用户可以通过 gcf 函数来获得当前图形窗口的句柄，或是在当前没有图形窗口的情况下创建一个新的窗口。

例如，在命令窗口中输入命令 get(gcf)(关于 get 函数的用法，用户可以参见本章第 13.2 节)，将在命令窗口中输出如下代码，由于当前状态下不存在图形窗口，因此，系统将产生

个新的图形窗口，如图 13-2 所示。

```
>> get(gcf)
 Alphamap = [ (1 by 64) double array]
 BackingStore = on
 CloseRequestFcn = closereq
 Color = [0.8 0.8 0.8]
 Colormap = [ (64 by 3) double array]
 CurrentAxes = []
 CurrentCharacter =
 CurrentObject = []
 CurrentPoint = [0 0]
 DockControls = on
 DoubleBuffer = on
 FileName =
 FixedColors = [ (3 by 3) double array]
 IntegerHandle = on
 InvertHardcopy = on
 KeyPressFcn =
 MenuBar = figure
 MinColormap = [64]
 Name =
 NextPlot = add
 NumberTitle = on
 PaperUnits = centimeters
 PaperOrientation = portrait
 PaperPosition = [0.634517 6.34517 20.3046 15.2284]
 PaperPositionMode = manual
 PaperSize = [20.984 29.6774]
 PaperType = A4
 Pointer = arrow
 PointerShapeCData = [ (16 by 16) double array]
 PointerShapeHotSpot = [1 1]
 Position = [232 248 560 420]
 Renderer = painters
 RendererMode = auto
 Resize = on
 ResizeFcn =
 SelectionType = normal
 ShareColors = on
 ToolBar = auto
 Units = pixels
 WindowButtonDownFcn =
 WindowButtonMotionFcn =
 WindowButtonUpFcn =
 WindowStyle = normal
```

```
WVisual = 03 (RGB 16 bits(05 06 05 00) zdepth 16, Hardware Accelerated, Opengl, Window)
WVisualMode = auto

BeingDeleted = off
ButtonDownFcn =
Children = []
Clipping = on
CreateFcn =
DeleteFcn =
BusyAction = queue
HandleVisibility = on
HitTest = on
Interruptible = on
Parent = [0]
Selected = off
SelectionHighlight = on
Tag =
Type = figure
UIContextMenu = []
UserData = []
Visible = on

>>
```

图 13-2　使用 gcf 函数产生的新的图形窗口

2. 核心(Core)对象

核心对象包括直线、文本和特殊对象(如光、图像和面图)等基本的绘图命令；而坐标系对象包括线、面图和等高线图等描述数据的对象。

表 13-2 列出了 MATLAB 7.0 语言中用户创建核心对象的函数。

表 13-2　MATLAB 7.0 语言中创建核心对象的函数

函数名	功能描述
axes	坐标轴
image	MATLAB 7.0 语言中的图像
light	光源
line	二维图形中最基本的图形对象
patch	按指定方式填充的多边形
rectangle	具有可设置边界和表面颜色的二维图形对象
surface	图形表面
text	图形中的文本

例 13-1　本例将创建 3 个图形对象，并给图形对象、坐标轴对象和面图命令设置特定的值，MATLAB 7.0 将给其他的属性设置默认值。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> [x,y] = meshgrid([-2:.4:2]);
>>Z = x.*exp(-x.^2-y.^2);
>>fh = figure('Position',[350 275 600 450],'Color','w');
>>ah = axes('Color',[.8 .8 .8],'XTick',[-2 -1 0 1 2],...
          'YTick',[-2 -1 0 1 2]);
>>sh = surface('XData',x,'YData',y,'ZData',Z,...
               'FaceColor',get(ah,'Color')+.1,...
               'EdgeColor','k','Marker','o',...
               'MarkerFaceColor',[.5 1 .85]);
>>
```

结果如图 13-3 所示。注意：此时 surface 函数并没有使用像高级功能函数 surf 那样的三维视角来显示图形，而仅仅是将新的对象添加到当前的坐标轴中，但是不改变坐标轴的属性，只改变了子对象的属性，包括新对象和坐标轴的上下限。

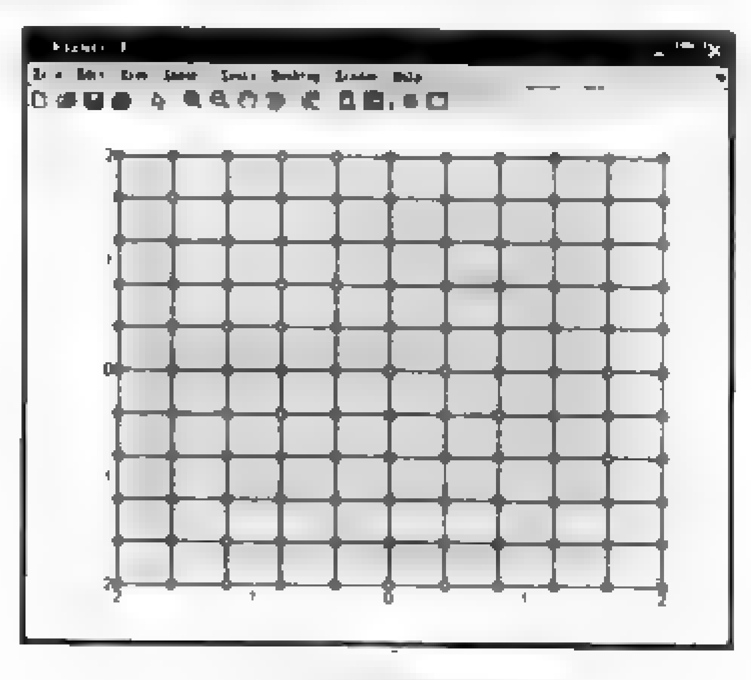


图 13-3　图形对象的创建

用户可以使用 view 函数改变图形的视角，在命令窗口中输入如下命令，并按 Enter 键确认，如图 13-4 所示。

```
>> view(3)
>>
```

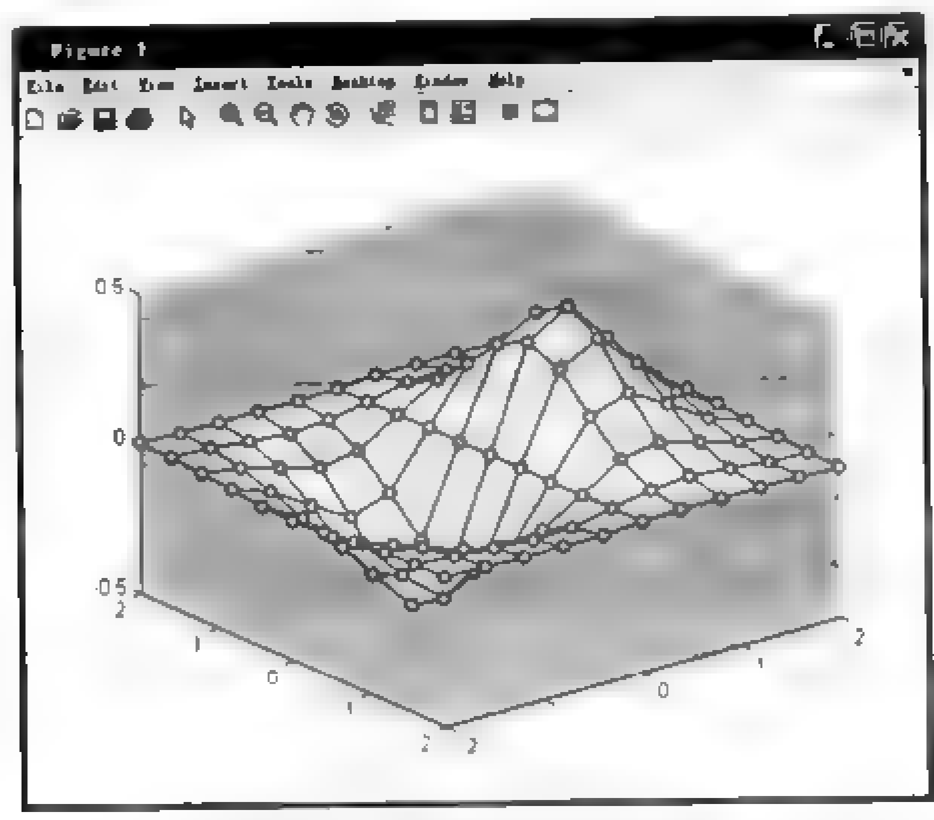


图 13-4　改变视角后的图形

3. 绘图(Plot)对象

MATLAB 7.0 提供了一系列的高级绘图函数来创建绘图对象，这些绘图对象的属性具有重要的意义，使用它们可以很方便地访问绘图对象所包含的核心对象的重要属性。

绘图对象的父对象可以是坐标轴对象或是群对象。表 13-3 列出了 MATLAB 7.0 中的绘图对象以及使用它们的绘图命令。

表 13-3　MATLAB 7.0 语言中的绘图对象及其绘图命令

函 数 名	功 能 描 述
areaseries	用于创建 area 图
barseries	用于创建 bar 图
contourgroup	用于创建 contour 图
errorbarseries	用于创建 errorbar 图
lineseries	提供给绘制线型图的 plot 和 plot3 等函数使用
quivergroup	用于创建 quiver 和 quiver3 图形
scattergroup	用于创建 scatter 和 scatter3 图形
stairseries	用户创建阶梯图形(stairs))
stemseries	用于创建 stem 和 stem3 图形
surfaceplot	提供给 surf 和 mesh 群函数使用

例 13-2　本例创建一个等高线图，然后修改该图的线型和线宽。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> [x,y,z] = peaks;
>> [c,h] = contour(x,y,z);
>>set(h,'LineWidth',3,'LineStyle',':')
>>
```

生成的图形如图 13-5 所示。

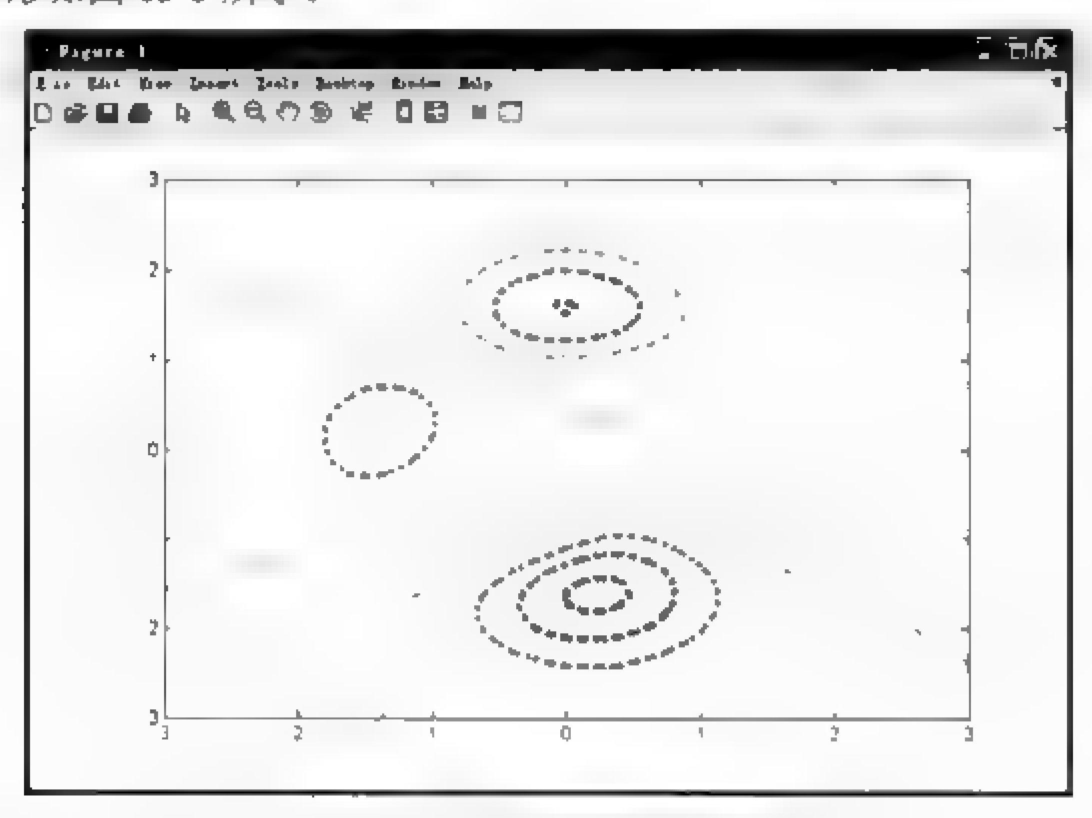


图 13-5　等高线图形的创建

4. 注释对象

用户一般习惯于使用图形编辑工具栏或是 Insert 菜单来创建注释对象，同时，用户也可以使用注释函数来创建注释对象。

注释对象在一个隐藏的坐标轴下创建，该坐标轴延伸到图形整个长度和宽度，这样，用户可以使法向坐标系(以图形的左下点为(0，0)，右上点为(1，1))在图形的任意部位定义图例注释。

例 13-3　使用注释矩形环绕多子图。

解：本例显示了如何创建注释矩形，并给图形中的两个子图高亮显示，本例还使用核心对象的属性 Position 和 TightInset 来决定注释矩形的位置和大小。

首先创建子图阵列，在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> x = -2*pi:pi/12 2*pi;
>> y   x.^2;
>> subplot(2,2,1 2)
>> plot(x,y)
>> h1  subplot(223);
>> y   x.^4;
```

```
>> plot(x,y)
>> h2=subplot(224);
>> y = x.^5;
>> plot(x,y)
>>
```

得到的图形如图 13-6 所示。

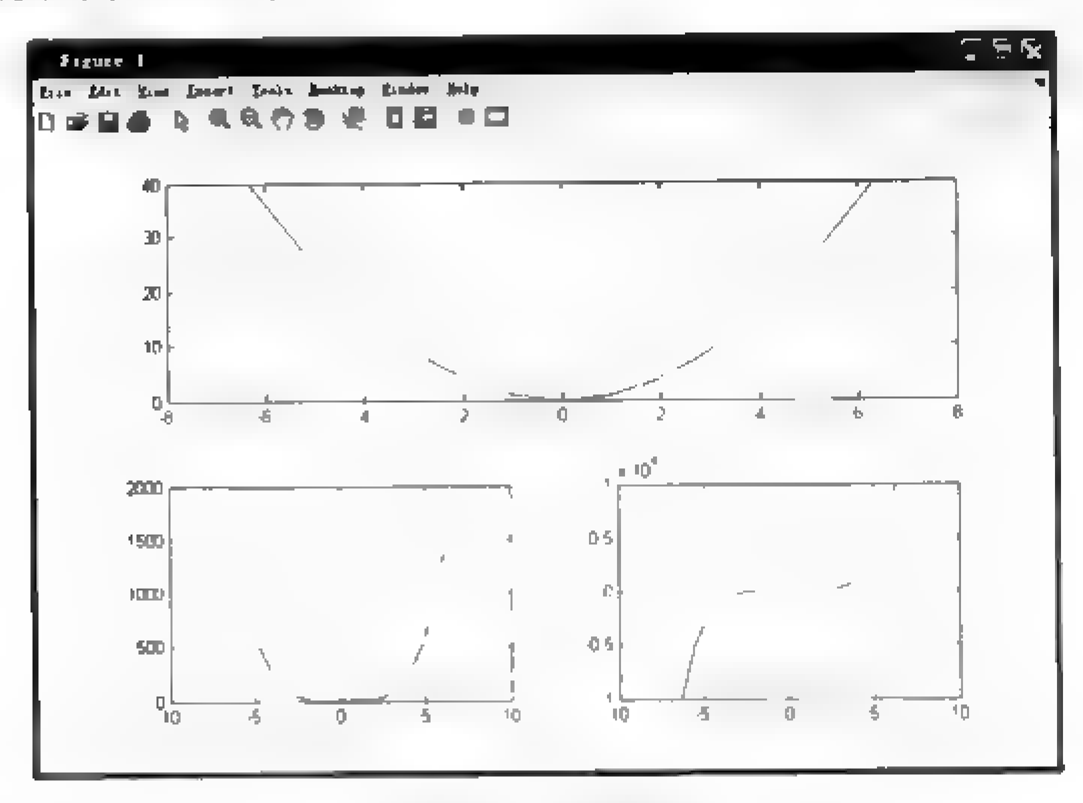


图 13-6　子图阵列

然后是定义注释矩形的位置和大小。在命令窗口中继续输入如下命令，并按 Enter 键确认。

```
>> p1 = get(h1,'Position');
>>t1 = get(h1,'TightInset');
>>p2 = get(h2,'Position');
>>t2 = get(h2,'TightInset');
>>x1 = p1(1)-t1(1); y1 = p1(2)-t1(2);
>>x2 = p2(1)-t2(1); y2 = p2(2)-t2(2);
>>w = x2-x1+t1(1)+p2(3)+t2(3); h = p2(4)+t2(2)+t2(4);
>>
```

得到的图形仍如图 13-6 所示。

最后再创建注释矩形来环绕下边的两个子图，并将矩形注释设置实性边界和红色填充色。命令窗口中继续输入如下命令，并按 Enter 键确认。

```
>> annotation('rectangle',[x1,y1,w,h],..
 'FaceAlpha',.2,'FaceColor','red','EdgeColor','red');
>>
```

运行程序的结果如图 13-7 所示。

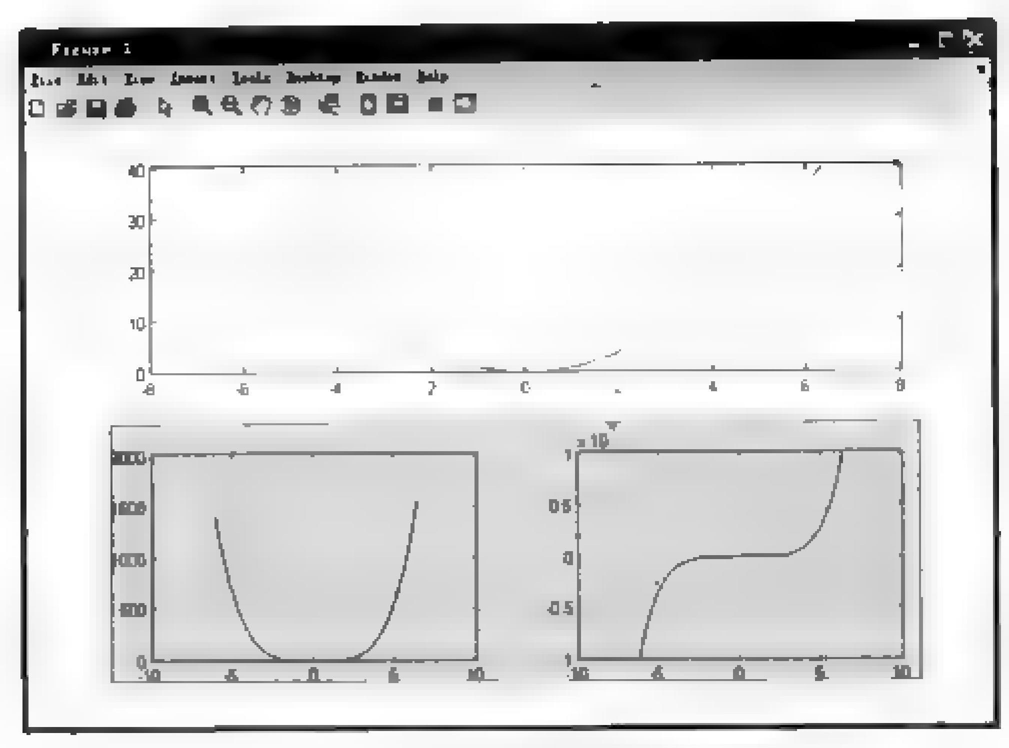


图 13-7 使用注释矩形环绕多子图

13.2 通用函数 get 和 set

所有对象都有属性来定义它们的特征，正是通过设定这些属性来修正图形显示的方式。尽管许多属性所有的对象都有，但与每一种对象类型(比如坐标轴、线和曲面)相关的属性列表都是独一无二的。对象属性可包括诸如对象的位置、颜色、类型、父对象和子对象等内容。每一个不同对象都有和它相关的属性，可以改变这些属性而不影响同类型的其他对象。和每一种对象类型(图形、坐标轴、线、文本、曲面、补片和图象)相关的完整的属性列表在本章的后面给出。

对象属性包括属性名和与它们相关联的值。属性名是字符串，它们通常按混合格式显示，每个词的开头字母大写，比如：LineStyle。但是，MATLAB 7.0 识别一个属性时是不分大小写的。另外，只要用足够多的字符来惟一地辨识一个属性名即可。例如，坐标轴对象中的位置属性可以用 Position、position、甚至是 pos 来调用。

当建立一个对象时，它用一组默认属性值，该值可以用两种方法来改变。可以用“{属性名，属性值}”对来建立对象生成函数；或者在对象建立后改变属性。前一种方法的例子是如下。

```
>> Hf_1=figure( 'color', 'white')
Hf_1 =
      2
>>
```

它用默认的属性值建立一个新的图形窗口，只是背景颜色被设为白色而不是默认的黑色。如图 13-8 所示。

图 13-8　建立图形对象

13.2.1　get 函数

在 MATLAB 7.0 中，使用 get 函数可以得到对象的属性及其属性值，其通常的调用格式为 *get(handle, ' PropertyName')*。关于它的详细使用方法，用户可以参见 get 函数的帮助信息。

例如，对图 13-7，可以使用 get 函数进行如下操作。

```
>>   p=get(Hf_1, 'position')                  % 位置
p =
    232     248     560     420
>> p=get(Hf_1, 'Children')                     % 子对象
p =
    Empty matrix. 0-by-1
>> p=get(Hf_1, 'color')                        % 颜色
p =
       1        1        1
>>get(Hf_1)                                     % 整个图形的属性
  Alphamap = [ (1 by 64) double array]
  BackingStore = on
  CloseRequestFcn = closereq
  Color    [1 1 1]
  Colormap = [ (64 by 3) double array]
  CurrentAxes = [230.002]
  CurrentCharacter =
  CurrentObject    [1]
  CurrentPoint    [553 476]
  DockControls = on
  DoubleBuffer = on
```

```
FileName =
FixedColors = [ (9 by 3) double array]
IntegerHandle = on
……
Type = figure
UIContextMenu = [151.003]
UserData = []
Visible = on
```

13.2.2　set 函数

在 MATLAB 7.0 中，使用 set 函数可以设置对象的属性值，其通常的调用格式如下。

- *set(H,'PropertyName',PropertyValue)*命令设置 PropertyName 的属性为 PropertyValue。
- *set(H,a)*命令中，a 为结构型变量，字段名为图形对象的属性名，字段值为映像的属性值。
- *set(H,pn,pv)*命令通过单元型变量为图形对象进行属性复制，其中 pn 和 pv 为单元型变量，pn 为 1×n 的字符型单元变量，各分量为图形对象的属性名，pv 可以是 m×n 的单元型变量，这里 m 为句柄数组 H 的长度，即 m=length(H).
- *set(H,'PropertyName1',PropertyValue1,'PropertyName2',PropertyValue2,...)*命令同时设置多个属性的值。

关于它的详细使用方法，用户可以参见 get 函数的帮助信息。

例如，可以在命令窗口中输入如下命令。

```
>>set(Hf_1, 'Position' , [232    248    560    420])
>>
```

此时图 13-7 变为如图 13-9 所示。

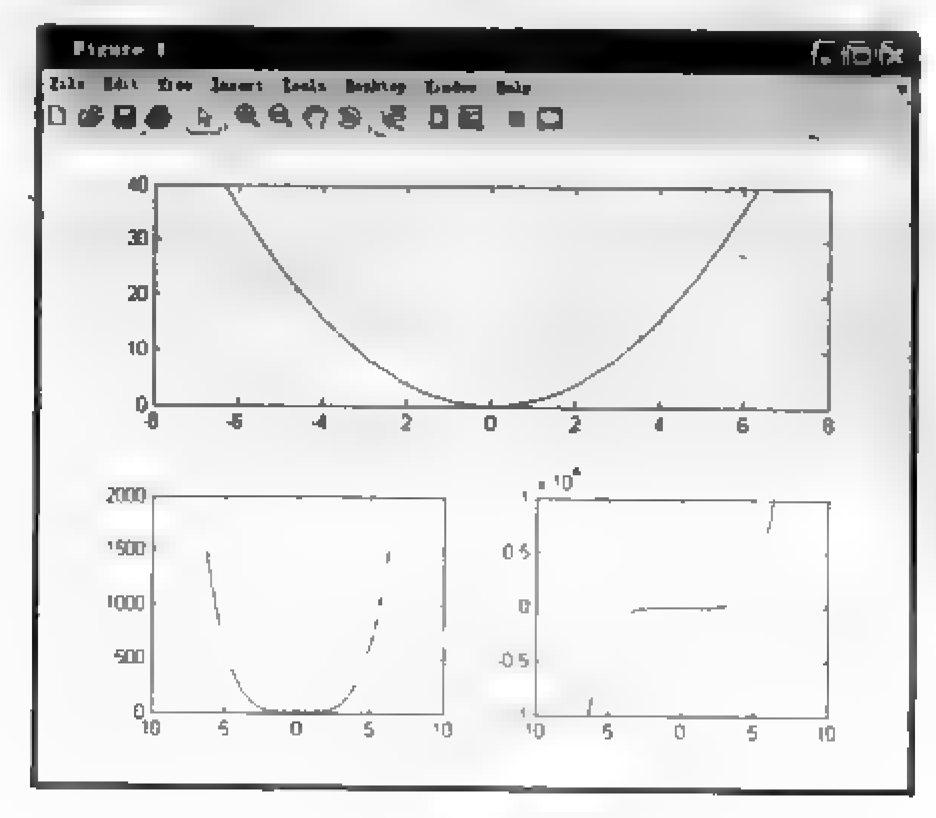


图 13-9　改变属性后的图 13-7

除了这些主要功能，函数 set 和函数 get 还能提供帮助。例如：*set(handle, 'PropertyName')* 返回一个可赋给由 handle 所描述对象的属性值列表。例如：

```
>>set(Hf_1, 'Units' )
[inches|centimeters|normalized|points|{pixels}]
>>
```

表明由 Hf_1 所引用的图形的 'Unites' 属性是 5 个可允许的字符串，而其中 'pixels' 是默认值。

如果指定一个没有固定值的属性，那么，MATLAB 7.0 就会通知如下：

```
>>set(Hf_1, 'Position' )
A figure's 'Position' property does not have a fixed set of property values.
>>
```

除了 set 命令，句柄图形对象创建函数(例如 figure，axis，line 等)接受多个属性名和属性值对。例如：

```
>> figure( 'Color', 'blue','NumberTitle', 'off', 'Name', 'My Figure')
>>
```

创建一个新的图形窗口，背景为蓝色，标有“My Figur”而不是默认标题“Figure No. 1”，如图 13-10 所示。

图 13-10　新建的图形窗口

为了形象说明上述概念，考虑下面的例子：

```
>> Hf_fig =figure   %  create a figure having an interger handle
 Hf_fig
      1
>> Hl_line=line   %  create a line having a floating-pointer handle
```

```
Hl_line =

   151 0020
>> set(Hl_line)        %   list settable properties and potential values
  Color
  EraseMode: [ {normal} | background | xor | none ]
  LineStyle: [ {-} | -- | : | -. | none ]
  LineWidth
  Marker: [ + | o | * | . | x | square | diamond | v | ^ | > | < | pentagram | hexagram | {none} ]
  MarkerSize
  MarkerEdgeColor: [ none | {auto} ] -or- a ColorSpec.
  MarkerFaceColor: [ {none} | auto ] -or- a ColorSpec.
  XData
  YData
  ZData

  ButtonDownFcn: string -or- function handle -or- cell array
  Children
  Clipping: [ {on} | off ]
  CreateFcn: string -or- function handle -or- cell array
  DeleteFcn: string -or- function handle -or- cell array
  BusyAction: [ {queue} | cancel ]
  HandleVisibility: [ {on} | callback | off ]
  HitTest: [ {on} | off ]
  Interruptible: [ {on} | off ]
  Parent
  Selected: [ on | off ]
  SelectionHighlight: [ {on} | off ]
  Tag
  UIContextMenu
  UserData
  Visible: [ {on} | off ]

>>   get(Hl_line)                      %   list properties and current property values
  Color = [0 0 0]
  EraseMode = normal
  LineStyle = -
  LineWidth = [0 5]
  Marker = none
  MarkerSize = [6]
  MarkerEdgeColor = auto
  MarkerFaceColor = none
  XData = [0 1]
  YData = [0 1]
```

```
	ZData = []

	BeingDeleted = off
	ButtonDownFcn =
	Children = []
	Clipping = on
	CreateFcn =
	DeleteFcn =
	BusyAction = queue
	HandleVisibility = on
	HitTest = on
	Interruptible = on
	Parent = [152.002]
	Selected = off
	SelectionHighlight = on
	Tag =
	Type = line
	UIContextMenu = []
	UserData = []
	Visible = on

>>
```

在上例中，所创建的线条中的 Parent 属性就是包含线条的坐标轴的句柄。而且所显示的图形列表被分为两组。在空行上的第一组，列出了该对象的独有属性，而空行下的第二组列出所有的对象共有的属性。函数 set 和函数 get 返回不同的属性列表：函数 set 只列出可以用 set 命令改变的属性；而 get 命令列出所有对象的属性。在上面的例子中，函数 get 列出了 Children 和 Type 属性，而 set 命令却没有。这一类属性只可读，但不能被改变，它们叫做只读属性。

与每一个对象有关的属性数目是固定的，但不同的对象类型有不同数目的属性。像上面所显示的，一个线条对象列出了 16 个属性，而一个坐标轴对象列出了 64 个属性。显然，透彻地说明和描述所有对象类型的全部属性超出本书的范围。但是，其中的很多属性本书以后要详细讨论，并且列出全部属性。

例 13-4　使用图像句柄绘制图形。

解：首先用非标准颜色画一条正弦曲线。线的颜色用 RGB[1 0.5 0]来指定，它是适中的橘黄色。在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> x=-2*pi:pi/40:2*pi;
>>y=sin(x);
>> Hl_sin=plot(x,y)
Hl_sin =
   151.0026
>>set(Hl_sin,'Color',[1 .5 0],'LineWidth',3)
>>
```

运行结果如图 13-11 所示。

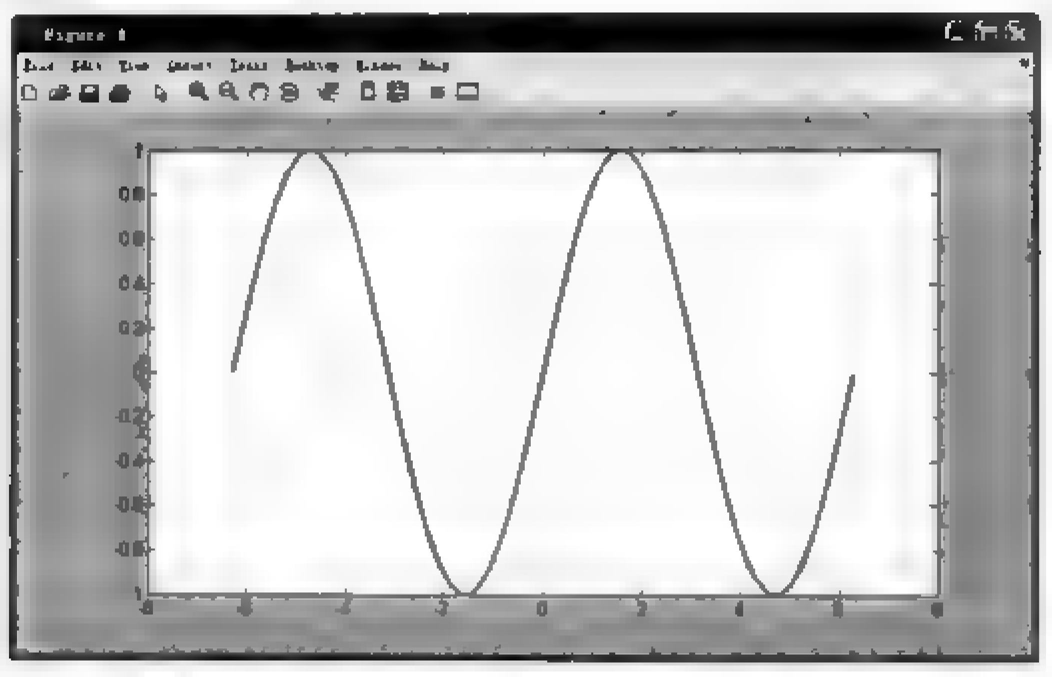


图 13-11 正弦曲线

现在再加一个浅蓝色的余弦曲线，继续在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> set(Hl_sin,'Color',[1 .5 0], 'LineWidth',3)
>>   z=cos(x);
>>   hold on
>>   Hl_cos=plot(x,z);
>> set(Hl_cos,'Color',[.75  .75  1])
>>
```

运行结果如图 13-12 所示。

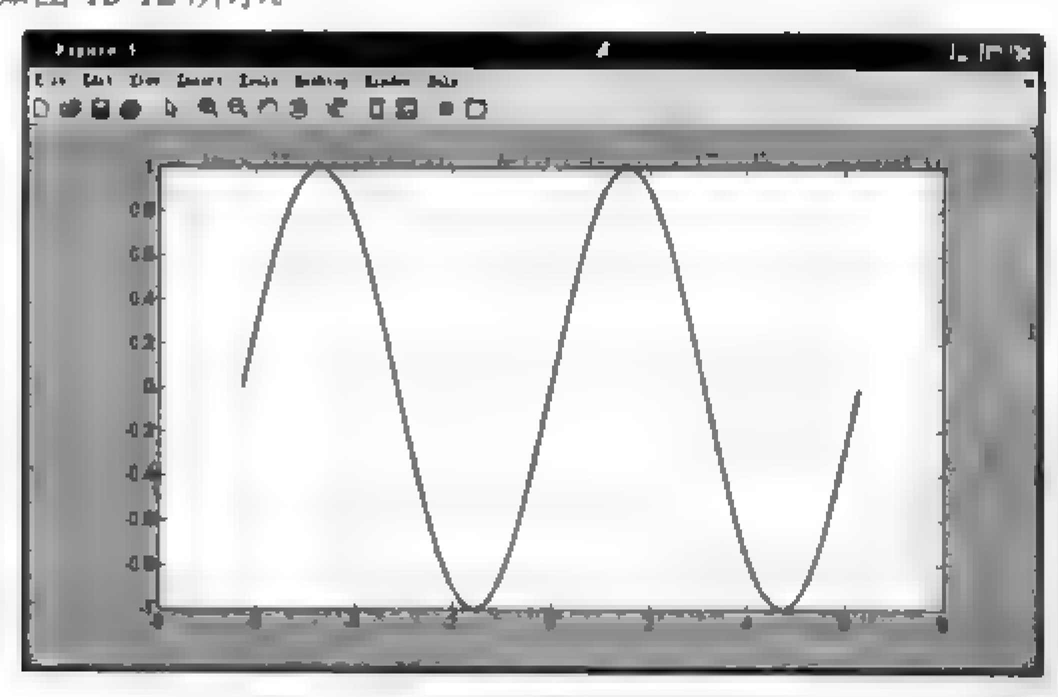


图 13-12 加上余弦曲线之后的图形

13.3 查 找 对 象

如前所述，句柄图形提供了对图形对象的访问途径，并且允许用函数 get 和 set 定制图

形。但是，如果用户忘记保存句柄或图形对象的句柄，或者当变量被覆盖时，如果要改变对象的属性，就必须进行对象句柄的查找。MATLAB 7.0 提供了查找对象的函数 gcf、gca、gco 和 findobj 等，前 3 个函数在表 13-1 中已经予以说明，下面对 3 个函数的使用方法进行更详细的介绍。

- *Hf_fig = gcf* 返回当前图形的句柄。
- *Hf_ax = gca* 返回当前图形的当前坐标轴的句柄。
- *Hx_obj = gco* 或是 *Hx_obj = gco(Hf_fig)* 获取当前对象。

而 findobj 函数用于寻找具有指定的属性值的对象，其使用形式如下。

- *Hx = findobj('P1name',P1value,...)* 命令返回根部和根部以下那些属性值与 *findobj* 参数对相匹配的对象句柄。
- *Hx = findobj(Objecthandle,'P1name',P1value,...)* 搜索限定在 *Objecthandle* 中列出的对象和它们的子对象。
- *H = findobj(Objecthandle,'flat','P1name',P1value,...)* 搜索限定在 *Objecthandle* 中列出的对象，不搜索它们的子对象。
- *H = findobj* 返回根对象和它所有的子对象的句柄。
- *H = findobj(Objecthandle)* 返回 *Objecthandle* 中列出的对象和它们的子对象的句柄。

当前对象的定义为用鼠标刚刚点过的对象。这种对象可以是除根对象(计算机屏幕)之外的任何图形对象。但是，如果鼠标光标处在一个图形中而未单击鼠标，gco 返回一个空距阵。为了让当前对象存在，必须选择一些东西。

一旦获得了一个对象的句柄，它的对象类型可以通过查询对象的 Type 属性来获得。该属性是一个字符串对象名，比如 figure、axes 或 text。例如：当需要一些除了 CurrentFigure、CurrentAxes 和 CurrentObject 之外的某些东西时，可以用函数 get 来获得一个对象的子对象的句柄向量。

```
>>Hx_kids=get(gcf, 'Children' )
```

返回一个向量，它包含当前图形子对象的句柄。

例 13-5　使用获得子对象 Children 句柄的技术彻底搜索句柄图形的层次结构中来找到所要的对象。

解：本例在画出一些数据后，寻找其中的绿色线条句柄。在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> x=-pi:pi/20:pi;
>> y=sin(x);z=cos(x);
>> plot(x,y,'r' ,x,z,'g');
>> Hl_lines=get(gca, 'Children');
>> for k=1:size(Hl_lines)
      Hl_green=Hl_lines(k);
      end
```

```
    end
>>Hl_green
Hl_green =
  153.0022
Hl_green =
  151.0038
>>
```

运行结果如图13-13所示。

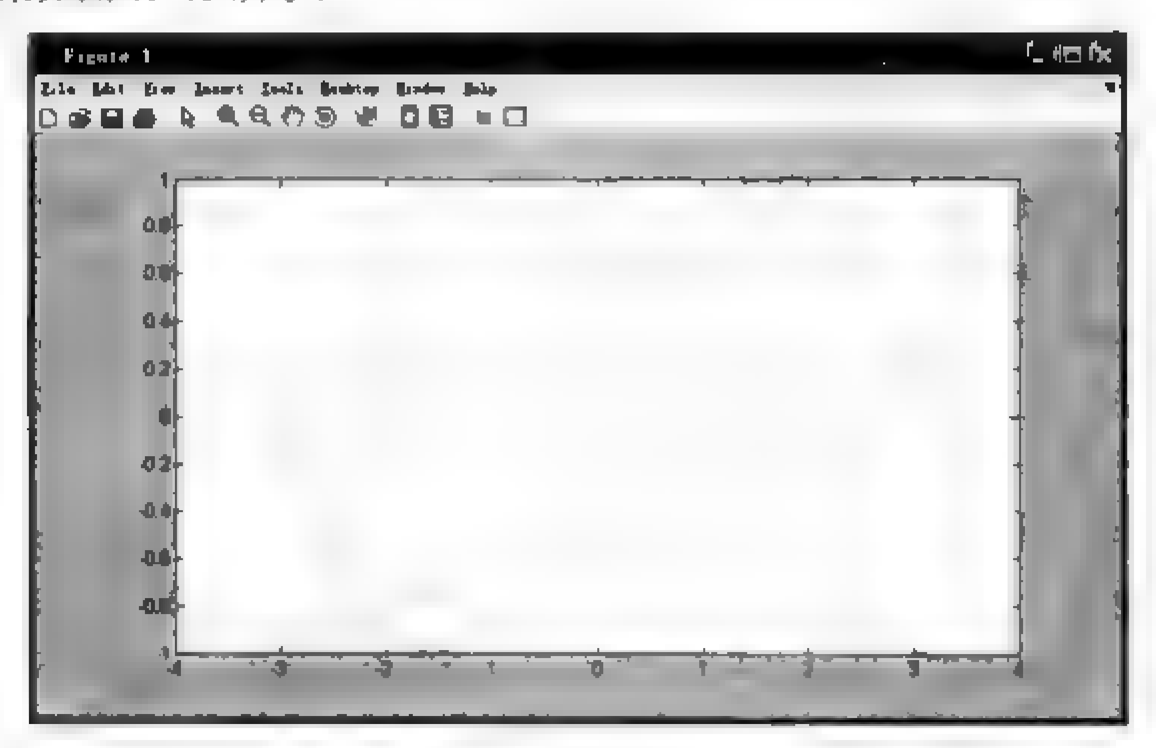


图13-13　寻找图形句柄

尽管这种技术有效，但是如果存在很多对象就变得复杂。该技术也丢失了标题和坐标轴标志中的文本对象，除非能逐个检测这些对象。

当有多个图形，每个图形上又有多个坐标轴时，可以使用如下程序查找所有绿色线条的句柄的问题。

```
>> Hf_all=get(0, 'Children' );   %   get all figure handles
>> for k=1:length(Hf_all)
>> Ha_all=[Ha_all,get(Hf_all(k), 'Children' )];   %   get all axes handles
>>end
>>for k=1:length(Ha_all)
>>   Hx_all=[Hx_all,get(Ha_all(k), 'Children' )];   %   get axes child handles
>>end
>>for k=1:length(Hx_all)
>>if get(Hx_all(k), 'Type' )== 'line'
>> Hl_all=[Hl_all;Hx_all(k)];   %   get line handles only
>>      end
>>      end
>> for   k=1:length(Hl_all)
>>     if get(Hl_all(k), 'Color' )==[0 1 0]
>>        Hl_green=[Hl_green,Hl_all(k)];   %   find green ones
```

```
>>     end
>> end
```

13.4 堆积次序

gco 命令返回当前对象的句柄，该对象为单击鼠标后的对象。那么，MATLAB 7.0 如何确定哪个对象被选中了呢？例如，当单击一幅图中两条线的交点时，应该返回哪个句柄？单击时指针离线有多远仍能选中该线？这些答案要取决于MATLAB 7.0选择对象规则和堆积次序。

堆积次序决定哪一对象叠加在其他对象上。开始时，堆积次序在对象被创建时就被决定，最后创建的对象在堆栈的顶部。例如，如果发出两条 figure 命令，就产生两个图形。第 2 个图画在第 1 个的上面，而最终的堆积次序是图 2 在图 1 的上面，当前图形 gcf 是图 2。如果发出 figure(1)命令，或者单击图 1，堆积次序就改变，图形框架 1 移动到堆栈的顶端，成为当前图形。

在前面的例子中，在计算机屏幕上，堆放次序可以从窗口交叠中显而易见。但是，情况并不总是这样的。如果画了两条线，在它们的交叉点，第二条线在第一条线的上面。如果用鼠标在第一条线其他一些点上单击，那么第一条线条就在栈顶，用鼠标单击交叉点会选中第二条线。当前的堆积次序是由一个给定的对象 Children 句柄出现的次序给出的。也就是说，$Hx_kids = get(handle,'children')$ 命令按堆积次序返回 children 句柄。存储句柄的向量 Hx_kids 的第一个元素在栈顶，而最后一个元素在栈底。

除了堆积次序，当用鼠标在一个对象附近单击时，该对象也会被选中。每一种对象类型都有与之相关的它自己的选择区域。例如，对于一条线来说，如果鼠标的光标在线条的 5 个像素之内，它就会被选中。曲面和文本对象的选择区域是包含对象的最小矩形。坐标轴对象的选择区域是坐标框本身加上标题和标志出现的区域。坐标轴内的对象，例如线条和曲面，在堆积次序中处于较高地位。因此，单击它们时会选择相关的对象而非坐标轴，而选择坐标轴选择区以外的区间就会选中图形本身。

13.5 默认属性

MATLAB 7.0 在建立对象时把默认属性赋给各对象。如果想不采用这些默认值，就必须使用句柄图形工具对它们进行设置。当每次都要改变同一属性时，MATLAB 7.0 允许设置用户自己的默认属性。MATLAB 7.0 让用户改变对象层次结构中任意一点上的单个对象或对象类型的默认属性。

可以使用一个特殊性质名字符串来设置默认值，该字符串以 Default 开头，跟之以对象类型和属性名。使用 set 命令中的句柄，确定对象父子等级图中的点，在该点使用默认

值，例如：

```
>>set(0, 'DefaultFigureColor',[.5  .5  .5])
```

将所有的新图形对象设为适中的灰色，而不是MATLAB 7.0默认的黑色。该属性值应用于根对象(它的句柄总是0)，所以所有新图形会有一个灰色的背景。

下面是另外一些可改变默认值的例子。

```
>> set(0, 'DefaultAxesFontSize' ,14)        %  larger axes fonts - all figures
>>set(gcf, 'DefaultAxesLineWidth' ,2)   %  thick axes lines - this figure
>> set(gcf, 'DefaultAxesColor' , 'y' )      %  yellow X-axis lines and labels
>> set(gcf, 'DefaultAxesGrid' , 'on' )      %  Y axis grid lines - this figure
>>set(0, 'DefaultAxesBox' , 'on' )          %  enclose axes - all figures
>>set(gcf, 'DefaultLineLineStyle' , ':' )   %  dotted linestyle - these axes
```

例13-6 本例创建一个具有白色背景的图形，并设置根对象下的坐标轴对象的默认值。

解：在命令窗口中输入如下命令，并按Enter键确认。

```
>> whitebg('w') %create a figure with a white color scheme
set(0,'DefaultAxesColorOrder',[0 0 0],
        'DefaultAxesLineStyleOrder','-|--|-.')
>> Z = peaks; plot(1:49,Z(4:7,:))
>>
```

运行结果如图13-14所示。

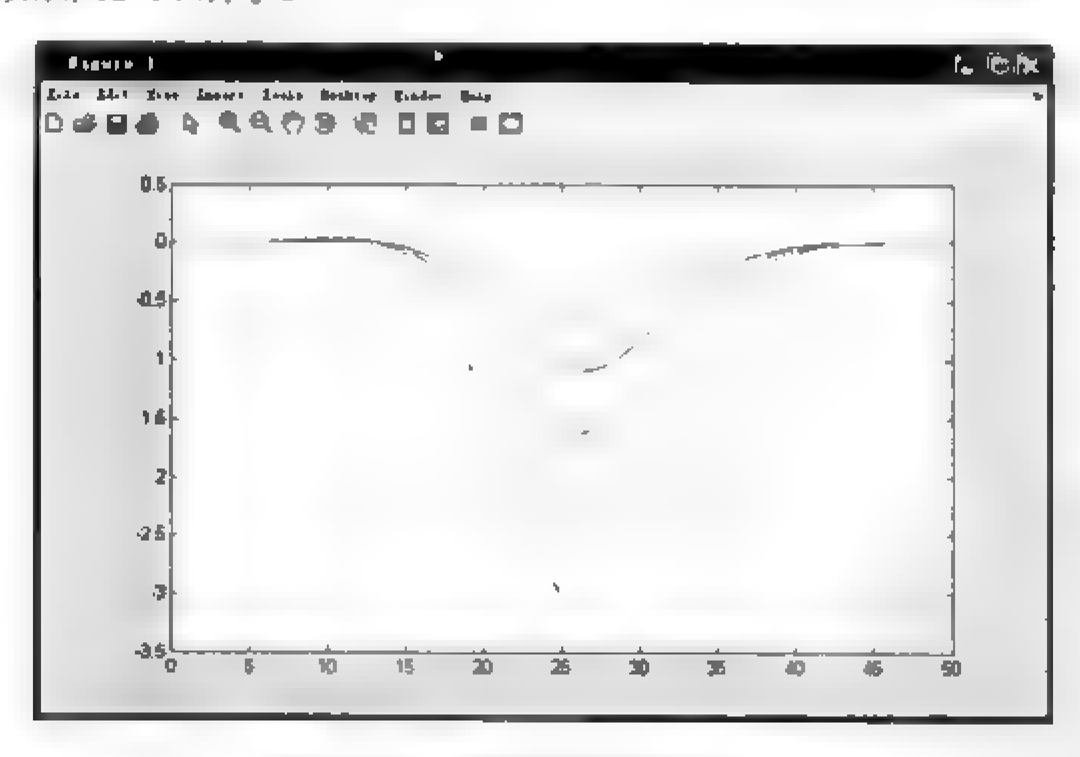


图13-14 设置图形的背景色为白色

13.6 习题

1. 简述句柄图形的意义以及句柄图形之间的父子关系。

2. 使用 MATLAB 7.0 的帮助系统学习图形对象的 Name 和 NumberTitle 属性，创建一个图形，绘制曲线 $y(x)=e^{x}$，其中 $-2 \le x \le 2$。修改前边所提的两个属性，使得图形的标题为“图形窗口”。

3. 编制一个程序，使得图形窗口的默认底色为桔黄色，默认的线宽为 4 个像素，并绘制图形 $\frac{x^2}{16}+\frac{y^2}{9}=1$。

4. 使用 MATLAB 7.0 的帮助系统学习 Axes 对象的 CurrentPoint 属性的使用方法，并使用该属性创建一个坐标轴对象，在该坐标轴内用线将连续的鼠标单击连接。使用 waitforbuttonpress 命令等待用户的鼠标单击，并在每次单击之后刷新图形，当按键盘上的任意键时终止程序。

5. 编制 MATLAB 7.0 程序，该程序绘制下面的函数：

$$\begin{cases} x(t)=\cos(\frac{t}{\pi}) \\ y(t)=2\sin(\frac{t}{2\pi}) \end{cases}，其中 -2<t\le 2。$$

该程序在绘制图形之后等待用户的鼠标输入，每单击其中的一条曲线，程序就随机修改该曲线的颜色，包括红色、绿色、蓝色、黑色和黄色。使用 waitforbuttonpress 命令等待用户的鼠标单击，并在每次单击之后刷新图形，使用 gco 函数来确认是哪个对象被选中，使用该对象的 Type 属性确认单击是否发生在曲线上。

第14章　创建图形用户界面GUI

用户界面是指用户与计算机或计算机程序的接触点或交互方式，是用户与计算机进行信息交流的方式。计算机在屏幕显示图形和文本，若有扬声器还可产生声音。用户通过输入设备，如：键盘、鼠标、跟踪球、绘制板或麦克风与计算机通讯等与计算机交流。用户界面设定了如何观看和如何感知计算机、操作系统或应用程序。通常是根据悦目的结构和用户界面功能的有效性来选择计算机或程序。

图形用户界面或GUI是包含图形对象，如：窗口、图标、菜单和文本的用户界面。以某种方式选择或激活这些对象，通常引起动作或发生变化。最常见的激活方法是用鼠标或其他设备去控制屏幕上的鼠标光标的运动。单击鼠标，标志着对对象的选择或其他动作。

生成用户图形界面可以实现以下几种功能：

- 编写一个需多次反复使用的实用函数，菜单、按钮、文本框作为输入方法具有意义；
- 编写函数或开发应用程序供别人使用；
- 创建一个过程、技术或分析方法的交互式示例；
- 简洁并且性能良好。

与上一章讨论MATLAB 7.0句柄图形功能的相同方式，它让用户按规定设计MATLAB 7.0显示信息的方法，本章所描述的图形用户界面的功能，它让用户定制用户与MATLAB 7.0的交互方式。命令窗口不是惟一与MATLAB 7.0的交互方式。

本章将介绍MATLAB 7.0中提供的图形用户界面特性。这些特性包括菜单、上下文菜单、按钮、滚动条、单选按钮、弹出式菜单和列表框等，并通过实例介绍如何编制GUI程序。

14.1　GUI对象层次结构

正如在上一章所展示的那样，由图形命令生成的每一事物是一个图形对象。图形对象不仅包括uimenu和uicontrol对象，而且还包括图形、坐标轴和它们的子对象。从另一个角度来看这一层次结构，计算机的屏幕本身是根结点，图形是根对象的子对象，坐标轴，uimenu，uicontrol是图形的子对象。

根可以包括多个图形，每个图形含有一组或多组坐标轴以及其子对象，每个图形也可以有一个或多个与坐标轴无关的uimenu和uicontrol。虽然uicontrol对象无子对象结点，但它们确实具有多种类型。uimenu对象常将其他的uimenu对象作为其子对象。如图14-1所示为GUI对象层次的结构图。

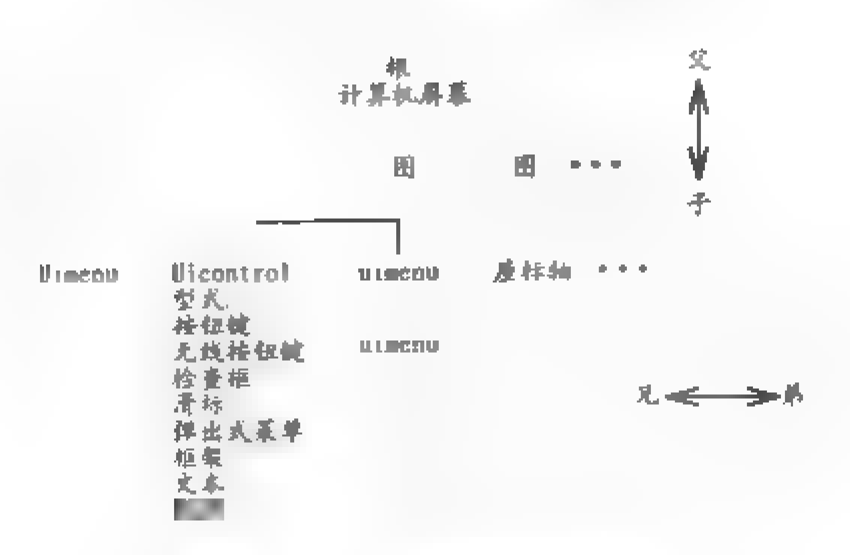


图 14-1　GUI 对象层次结构图

运行 MATLAB 7.0 的不同型号的计算机或平台上，产生不同的图形显示。Unix 工作站使用不同的 X Window 系统，具有几个窗口程序，如 mwn 或 twm 以控制显示的布局。PC 机靠 Microsoft Windows 或 Windows NT 进行窗口管理，Macintosh 计算机用 Macintosh 工具箱程序作窗口。虽然在各种平台上，显示看起来有很大的不同，但大多数情况下，句柄图形的编码是一致的。MATLAB 7.0 在内部处理平台和窗口系统的差别。体现句柄图形例程的函数，包括应用 uimenu 和 uicontrol 对象的函数，通常运行在所有平台。

14.2　GUI 的基本知识

本节主要介绍如何新建 GUI、打开 GUI、Layout 编辑器以及 GUIDE 模板等关于 GUI 的基础知识。

14.2.1　启动 GUI

用户欲启动 GUI 操作，可以在命令窗口中输入 guide 命令，此时将启动 GUIDE Quick Start 对话框，如图 14-2 所示。

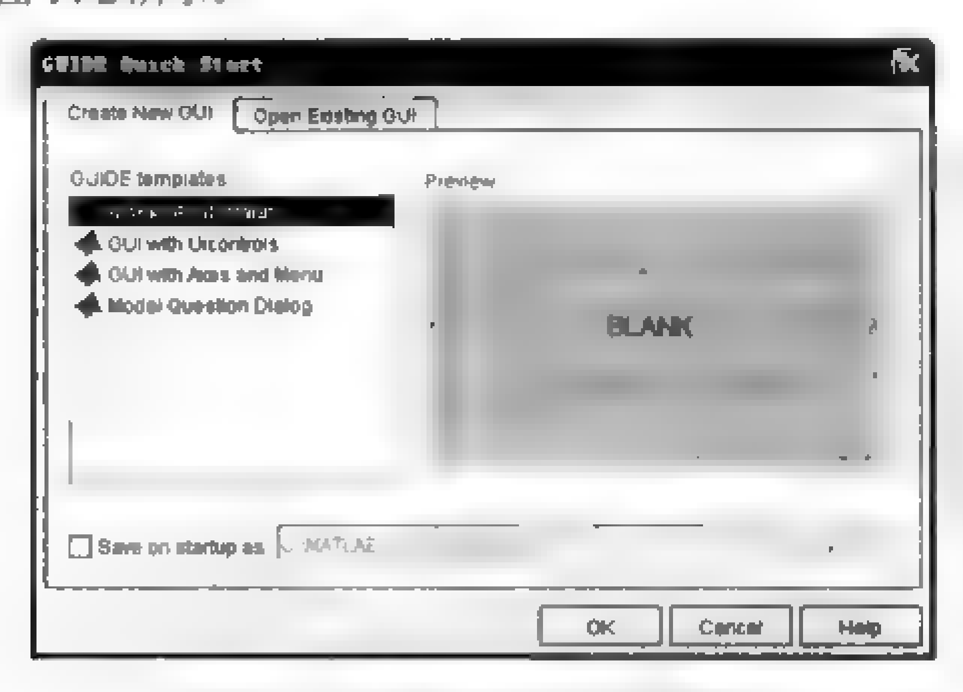


图 14-2　GUIDE Quick Start 对话框

利用 GUIDE Quick Start 对话框，用户可以进行如下操作：

- 利用 GUIDE 模板创建一个新的 GUI——预先创建 GUIs，使得用户可以依照自己的目的进行改进。
- 打开已有的 GUI。

一旦用户做出了上述之一的选择，单击 OK 按钮将在 Layout 编辑器中打开 GUI。

14.2.2　布局(Layout)编辑器

当用户在 GUIDE 中打开一个 GUI 时，该 GUI 将显示在 Layout 编辑器中，Layout 编辑器是所有 GUIDE 工具的控制面板。如图 14-3 所示为一个空白 GUI 模板的布局(Layout)编辑器。

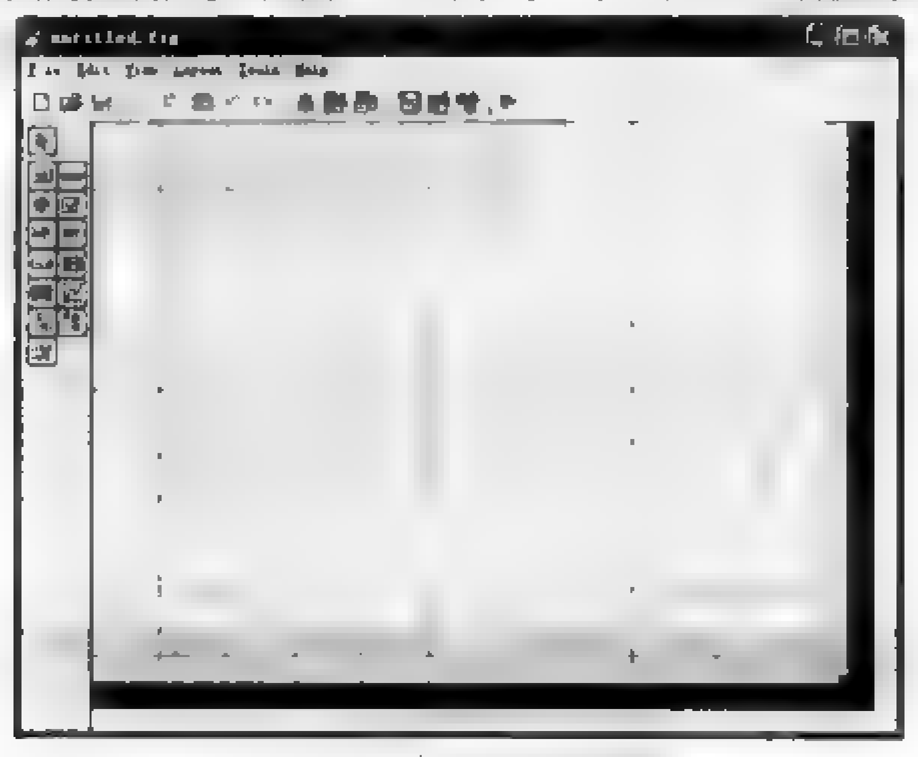


图 14-3　空白的 GUI 模板

用户可以使用鼠标拖动模板左边的控件(按钮、坐标轴、单选按钮等)到中间的布局区域，例如，当用户拖动一个按钮到布局区域时，将产生如图 14-4 所示的效果图。

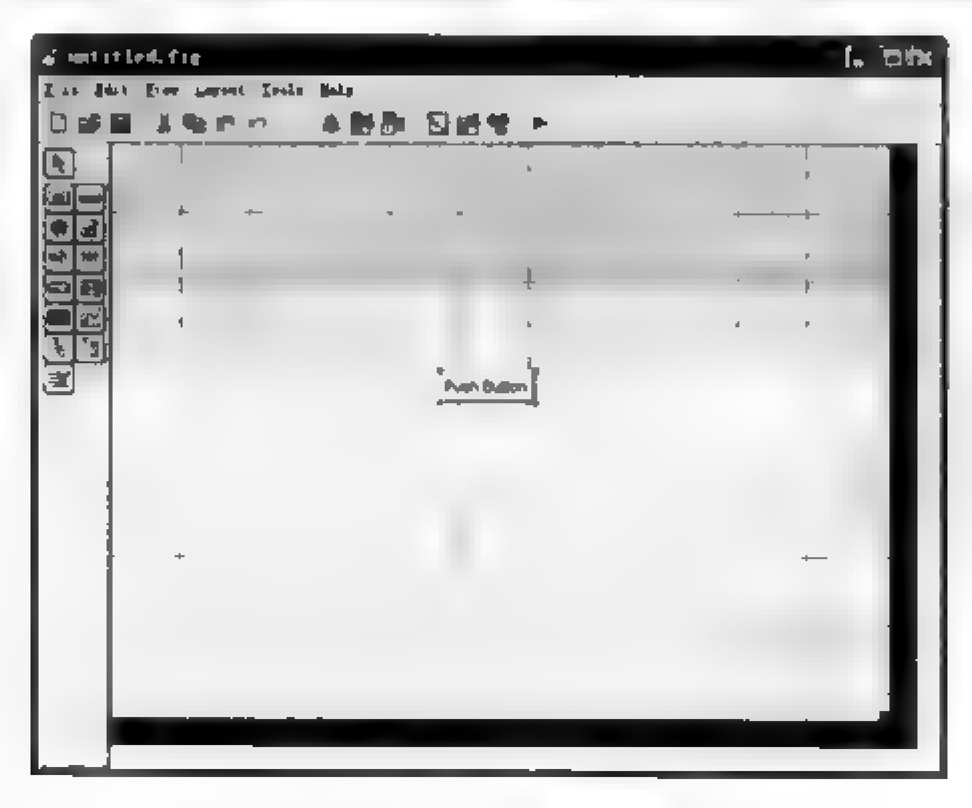


图 14-4　给 GUI 添加按钮

14.2.3　GUIDE 模板介绍

GUIDE Quick Start 对话框提供了几种常用的 GUI 模板。使用 GUI 模板的好处是与使用空白 GUI 相比，用户可以更方便地改变该模板的布局和功能。一旦用户选择了其中的一种模板，在 GUIDE Quick Start 对话框的右侧就出现该模板的预览。例如，用户选择了 GUIDE with Uicontrols 模板，在对话框的右边将产生如图 14-5 所示效果。

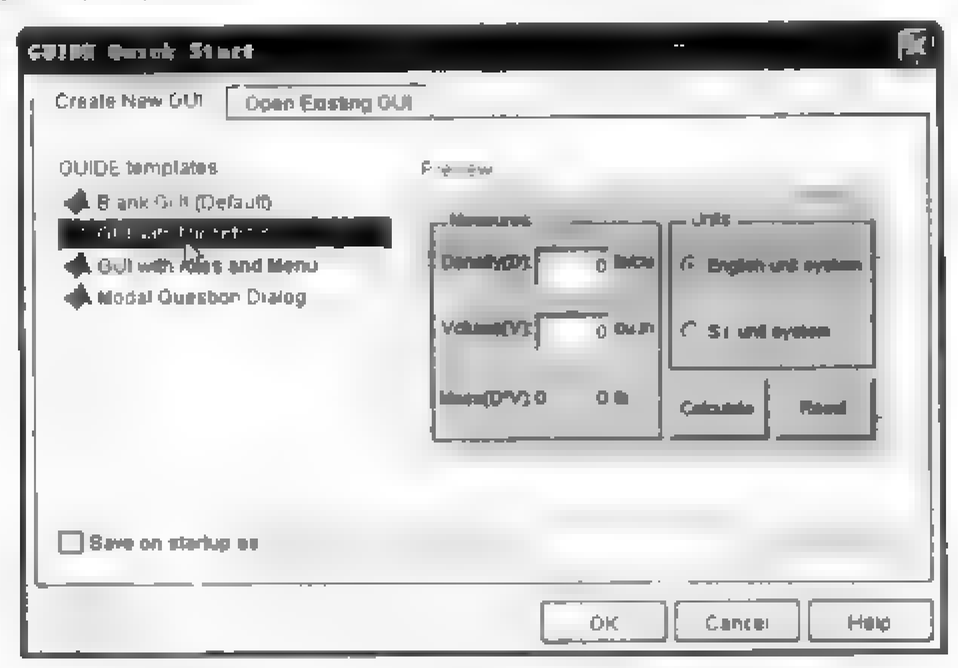


图 14-5　GUIDE with Uicontrols 模板预览图

单击 OK 按钮将在布局编辑器中打开该模板，如图 14-6 所示。

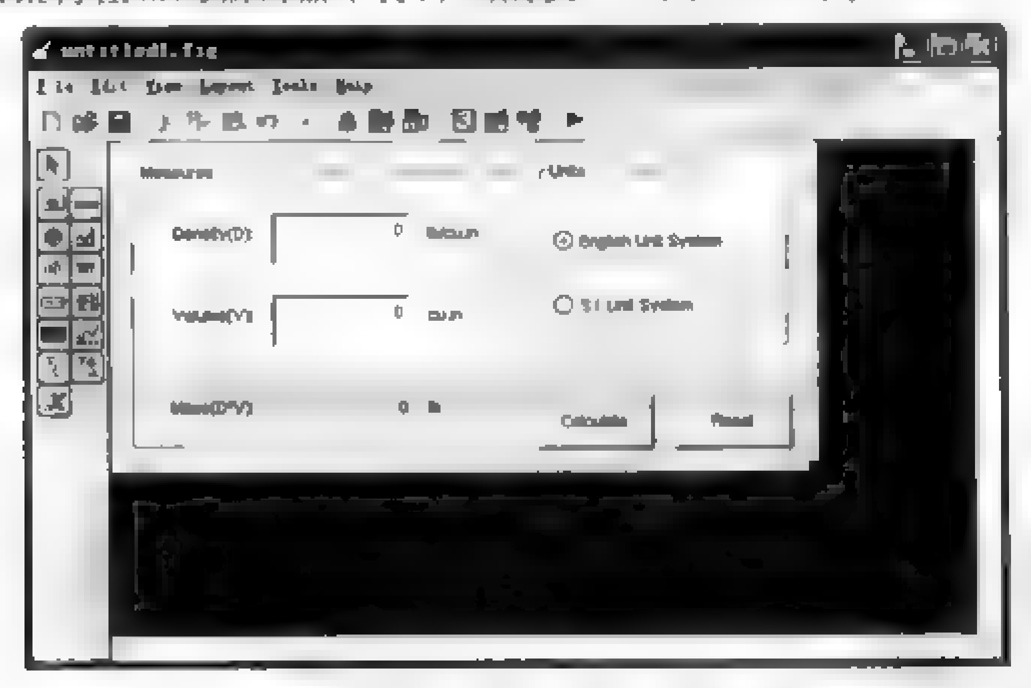


图 14-6　GUIDE with Uicontrols 模板

14.2.4　运行 GUI

单击工具栏最右边的绿色按钮，即运行当前的 GUI 窗口，如图 14-7 所示。

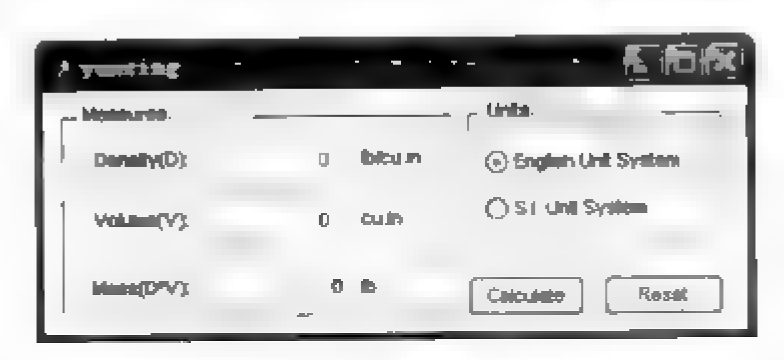


图 14-7　运行 GUI

如果是第一次运行，那么 MATLAB 7.0 首先将提示对该 GUI 窗口进行保存，并在运行的同时弹出 M 文件给用户进行编辑操作。

14.3　创建 GUI 对象

本节将介绍如何创建 GUI 对象，包括 GUI 窗口的布局、GUI 控件属性的设置以及 GUI 的编程。

14.3.1　GUI 窗口的布局

按照 14.1.1 节所介绍的那样，启动 GUI 之后，用户及可以进行 GUI 窗口的布局，包括改变 GUI 窗口的大小、给 GUI 窗口增加控件和对控件进行对齐操作。下面对这些操作分别予以介绍。

1. 改变 GUI 窗口的大小

在布局编辑器中可以很方便地改变 GUI 中网格区域的大小，只需单击网格区域的右下角，当鼠标变为箭头形式时，拖动鼠标，即可适时改变窗口的大小，如图 14-8 所示。

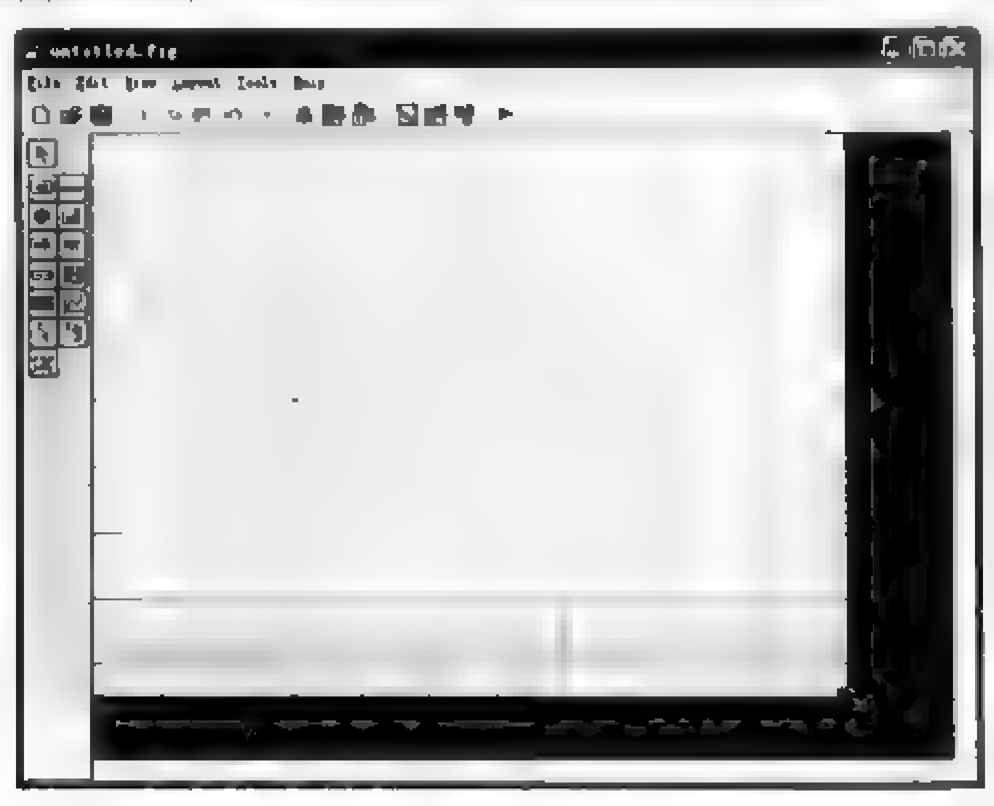


图 14-8　改变 GUI 窗口的大小

如果用户想精确地改变 GUI 窗口的大小和位置，可以进行如下操作。

- 从 View 菜单中选择 Property Inspector 项。
- 选择 Units 选项后边的按钮并从下拉菜单中选择 Centimeters 选项，如图 14-9 所示。
- 单击 Position 项前的“+”，如图 14-10 所示，其中 x 和 y 的坐标代表 GUI 窗口左下角的位置，width 和 height 代表 GUI 窗口的尺寸。
- 此时即可以改变 x 和 y 的坐标以及 GUI 窗口的尺寸。
- 重新设置 Units 选项后的单位为 Characters。

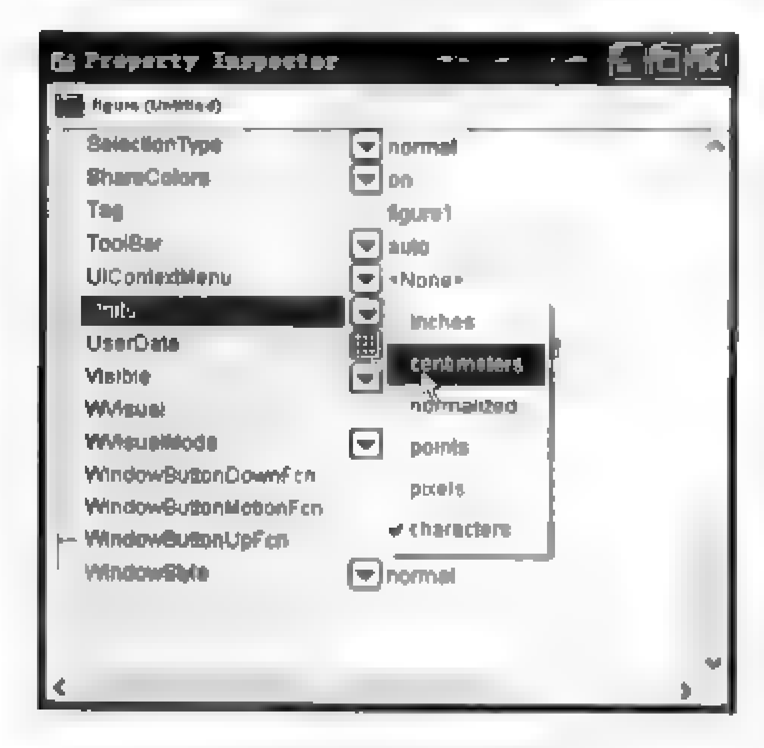


图 14-9　Centimeters 选项

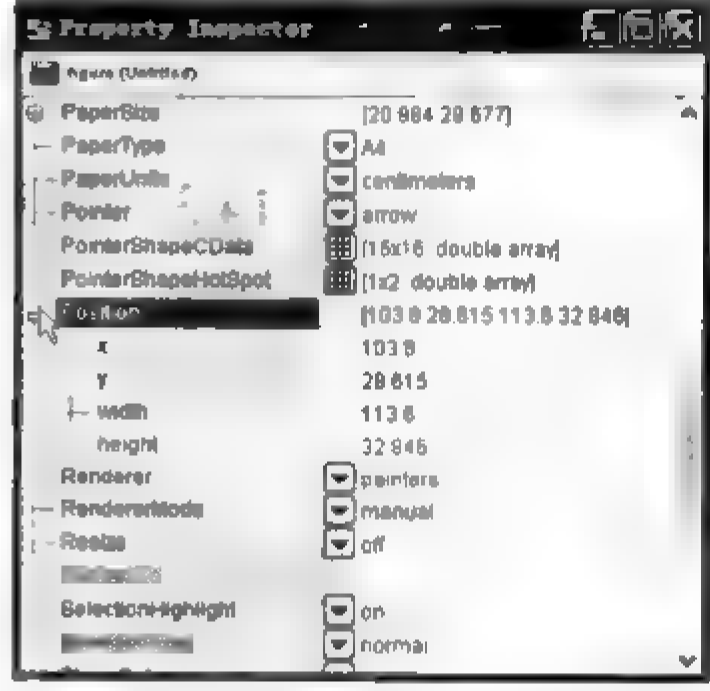


图 14-10　GUI 窗口大小和尺寸的调整

2. 控件的添加和对齐

在 14.1.2 节中已经介绍了控件的添加方法，下面举例介绍一下控件的布局，首先打开一个空白的 GUI 模板，并在其布局编辑器中添加控件，如图 14-11 所示。

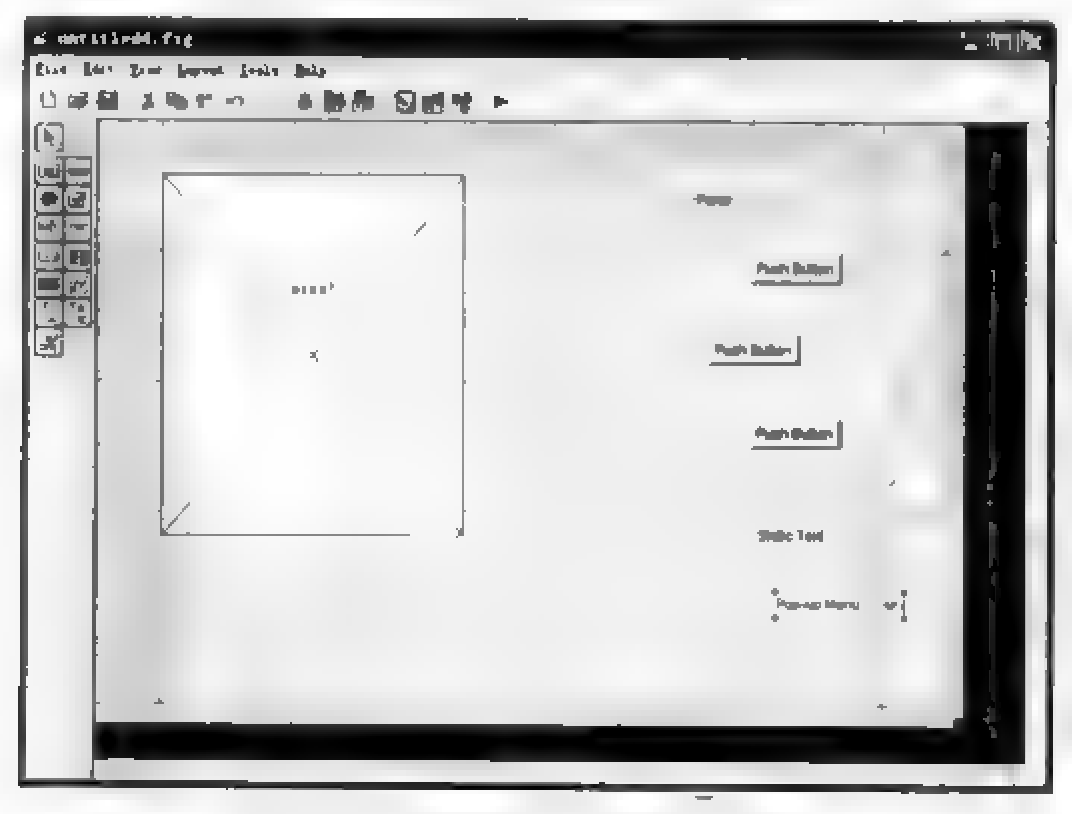


图 14-11　添加控件之后的 GUI 窗口

然后对图 14-11 中的控件进行对齐操作，单击工具栏上的对齐按钮 ，系统将弹出

Align Objects 对话框，如图 14-12 所示，用户可以使用它很方便地对 GUI 窗口中的控件进行对齐操作。但是，必须要保证进行对齐操作的控件有共同的父对象。

例如，如果用户需要对齐图 14-11 中 panel 板中的 3 个 Push Botton，只需进行如下操作：

(1) 按下 Ctrl 键，单击选中此 3 个按钮；

(2) 在工具栏中选择对齐按钮 ，弹出 Align Objects 对话框；

(3) 进行如图 14-13 所示的操作，将每个按钮之间的垂直距离设为 35 个像素，水平方向采用左对齐的方式；

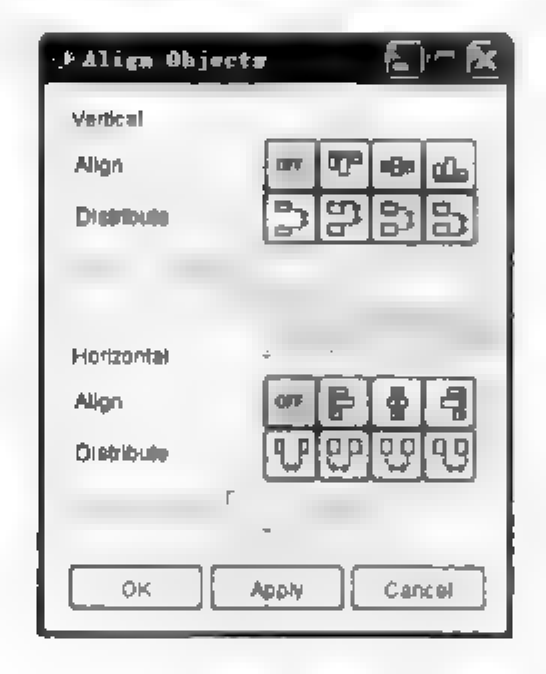


图 14-12　Align Objects 对话框

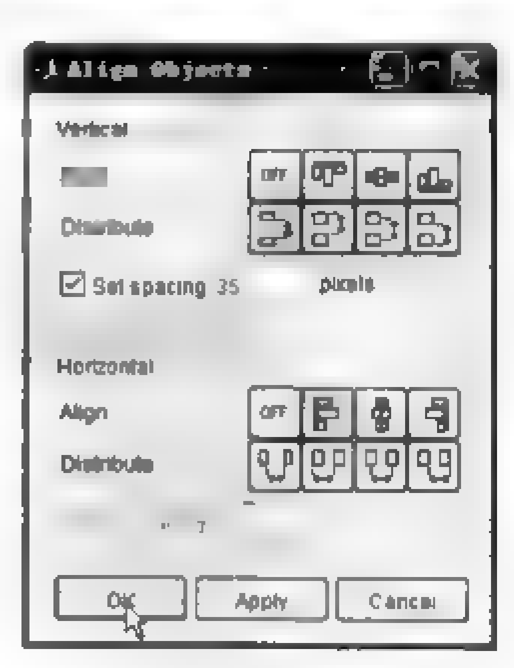


图 14-13　对齐操作

(4) 单击 OK 按钮确认。

对齐后的效果如图 14-14 所示。

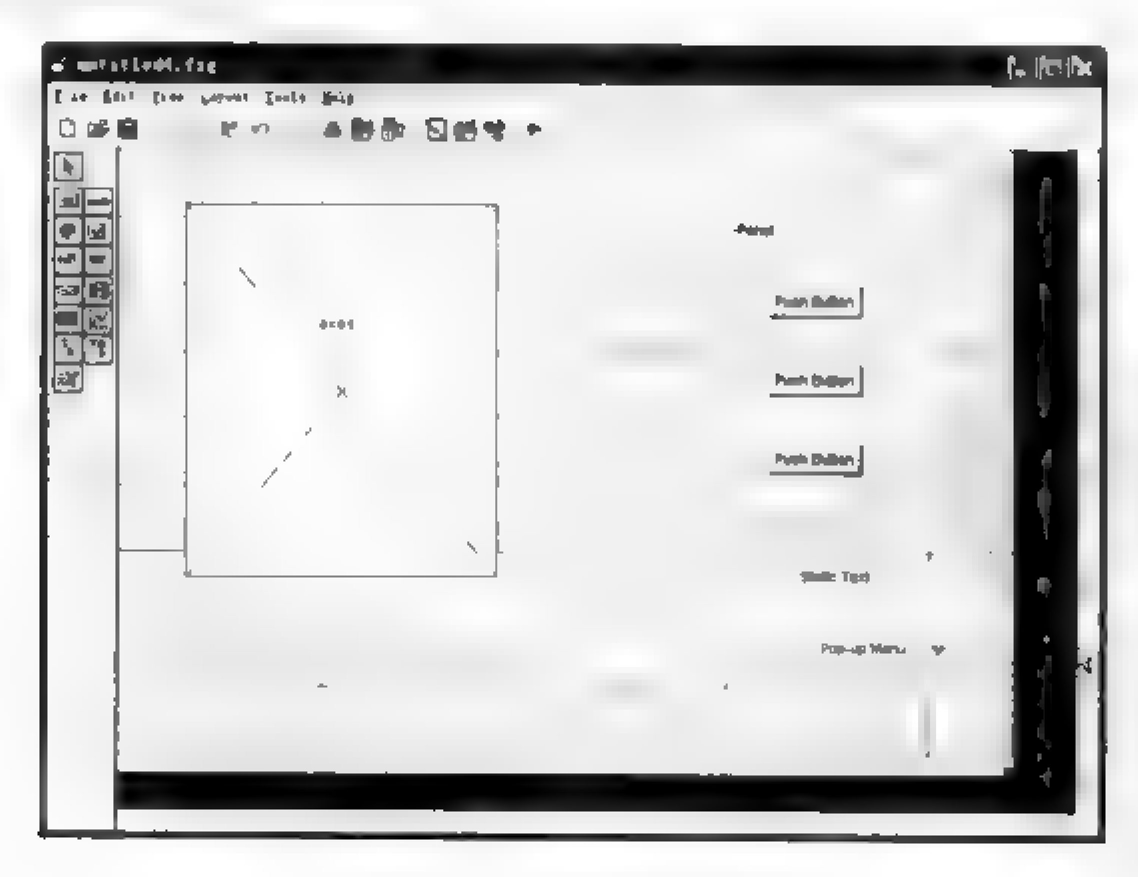


图 14-14　对齐之后的 GUI 窗口

14.3.2　GUI 控件的属性控制

用户可以使用属性查看器(Property Inspector)来设置布局窗口中控件的属性，Property Inspector 提供了一系列可以设置的属性并显示其当前值。下面将对 GUI 控件的属性设置进行简单介绍。

1. 属性查看器的显示

用户可以使用如下 3 种方式打开 Property Inspector。

- 在布局窗口中双击某个控件。
- 在 View 菜单中选择 Property Inspector 选项。
- 右击某个控件并从弹出的快捷菜单中选择 Inspect Properties 选项。

上述 3 种操作都将调出属性查看器，如图 14-15 所示。

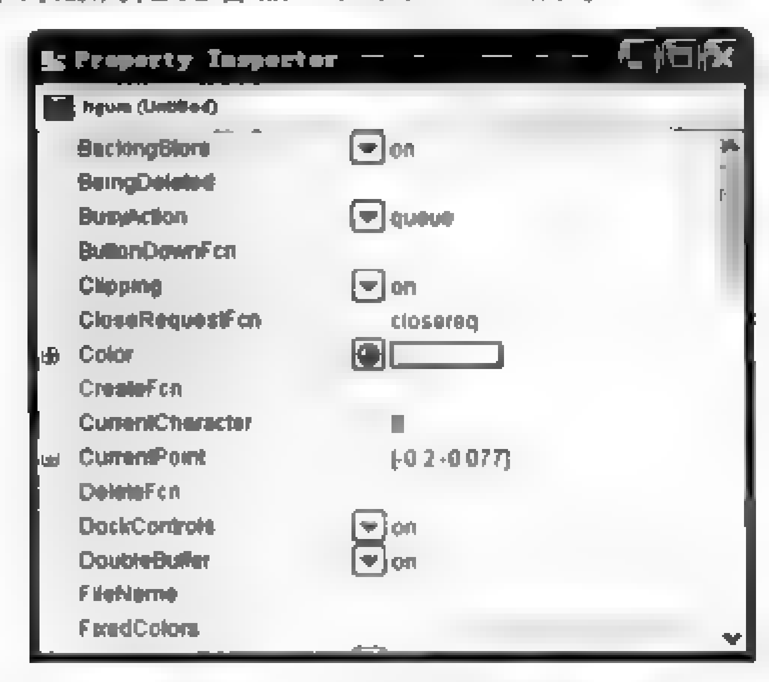


图 14-15　Push Botton 的属性查看器

2. 一些常用的属性

下面介绍 GUI 控件的一些常用属性，有关 GUI 控件的详细属性，用户可以在打开属性查看器之后进行实际操作的时候慢慢体会。

(1) 标签(Tag)属性

标签属性相当于控件的身份证明。GUIDE 会自动给用户添加给布局编辑器的每一个控件赋予一个标签值(例如 pushbutton1)，然后，利用这个值来命名与响应(Callback)属性相关的相应操作。

标签值清晰地描述该控件的特征，这样在编制 M 文件时就可以很方便地对各个控件加以区分。例如，如果用户有意将一个按钮设为该 GUI 的关闭按钮，可以将该按钮的标签值设为 close_button。

GUIDE 通常使用 Tag 属性进行如下操作：

- 在运行和保存 GUI 时给产生的响应创立名字，如 close_button_Callback。
- 给响应设置相应的响应属性。

- 给包含对象句柄的句柄结构增添一个域，如 handles.close_button。

(2) 响应(Callback)属性

用户在执行 GUI 的 M 文件时，响应属性标明被激活的某个控件的响应。

(3) 字符(String)属性

字符属性是指控件表面显示的名字，例如：

- 对 buttons、check boxes、list boxes、edit text 和 static text 等控件来说，字符属性在控件的上边或是附件显示。
- 对 edit text 控件来说，字符属性是指在下拉列表框中显示的一系列的字符串。当用户对文本进行编辑时，字符属性将自动更新。

(4) 数值(Value)属性

数值属性指控件包含的一个数字值，该数字值必须在 Max 和 Min 属性的范围内，例如：

- 对 radio button 和 check boxe 控件来说，Max 和 Min 的默认值分别设为 1 和 0。当 radio button 或 check boxe 被选中时，值为 1，否则为 0。
- 当用户拖动 slider 控件时，数值属性的值设置为 Max 和 Min 中的某个值，具体的大小由滑动条的相对位置决定。

14.3.3　菜单的添加

在每一个窗口系统中使用菜单让用户选择命令和选项。通常在显示屏或窗口的顶部有一菜单条。移动鼠标光标到菜单标志上单击，顶层菜单就被选中，在顶层菜单上单击，则打开下拉菜单。然后，移动鼠标光标至下拉菜单项并再次单击鼠标，选择菜单项，并在下拉菜单中选择相应的操作。

一个菜单项还可用自己的菜单项列表作为子菜单。子菜单项在子菜单的标志右边显示小三角或箭头以表示菜单还有更多子菜单项可供选择。如果子菜单的菜单项被选择，另一个具有更多菜单项的菜单显示在此菜单的右边的下拉菜单中，选中其中一个菜单项也引起某些动作的产生。

子菜单可以嵌套，但层次的数目受到窗口系统及有用资源的限制。

MATLAB 7.0 提供了两种类型的菜单形式。

- 主菜单：这种菜单的标题显示在图形窗口的顶部。
- 弹出式菜单对象：当用户右击某个图形对象时弹出。

用户可以单击工具栏上的图标 图标来创建这两种菜单，如图 14-16 所示。

图 14-16　菜单编辑器

1. 主菜单的创建

用户在创建了主菜单之后，MATLAB 7.0 将该菜单的标题添加到主菜单栏上，此时用户即可给该菜单添加菜单项，每个菜单项都可以包含一个子菜单，而子菜单也可以有自己的子菜单。

(1) 菜单属性的设置

单击图 14-16 中的菜单标题 Untitled 1，将在菜单编辑器的右边显示该菜单的属性提供给用户进行编辑，如 Label、Tag、Accelerator、Separator 和 Checked 等属性。单击 More options 按钮将显示更多的菜单属性，而 View 按钮是一个对响应进行编辑的子函数。修改菜单的部分属性后，所得图形如图 14-17 所示。

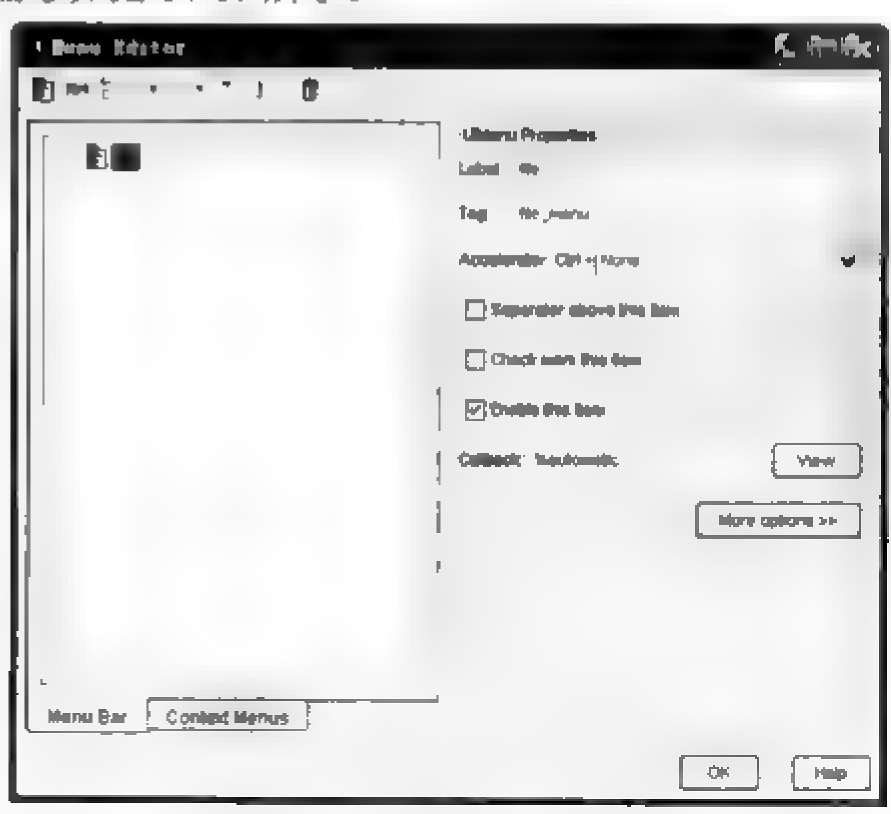


图 14-17　菜单属性的设置

(2) 给菜单增添菜单项

用户可以使用工具栏上的 New Menu Item 图标给当前菜单增添菜单项，如图 14-18 所示。

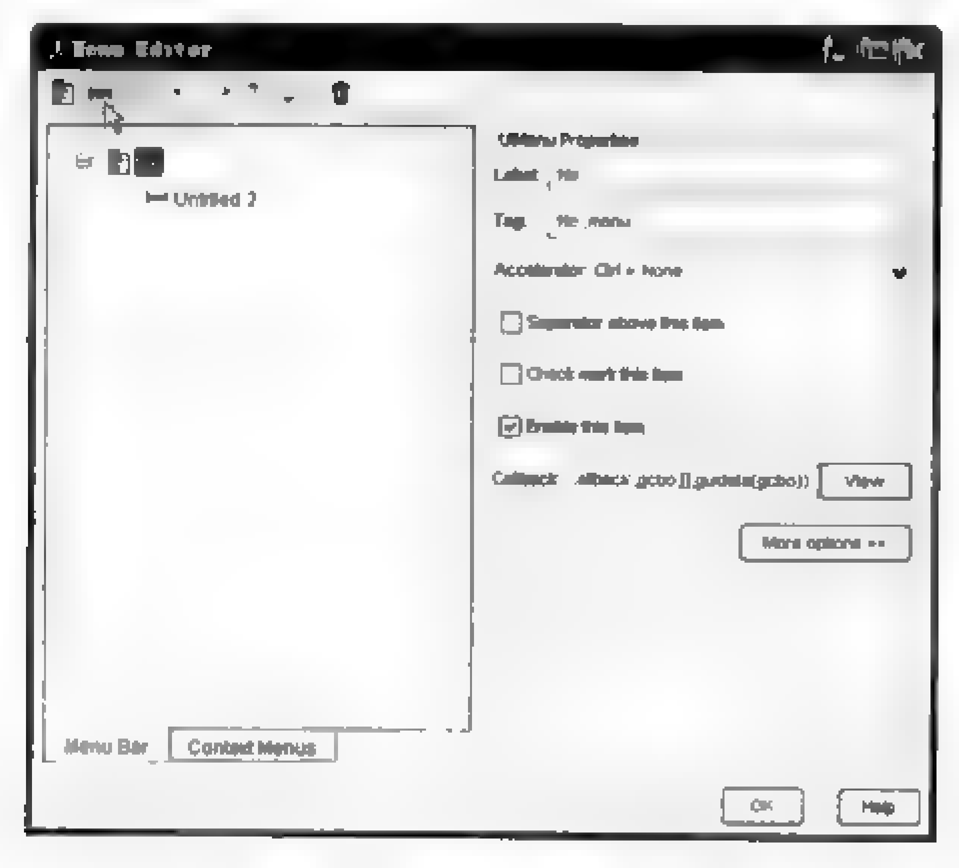


图 14-18　给菜单增添菜单项

创建完成之后的菜单编辑器如图 14-19 所示。

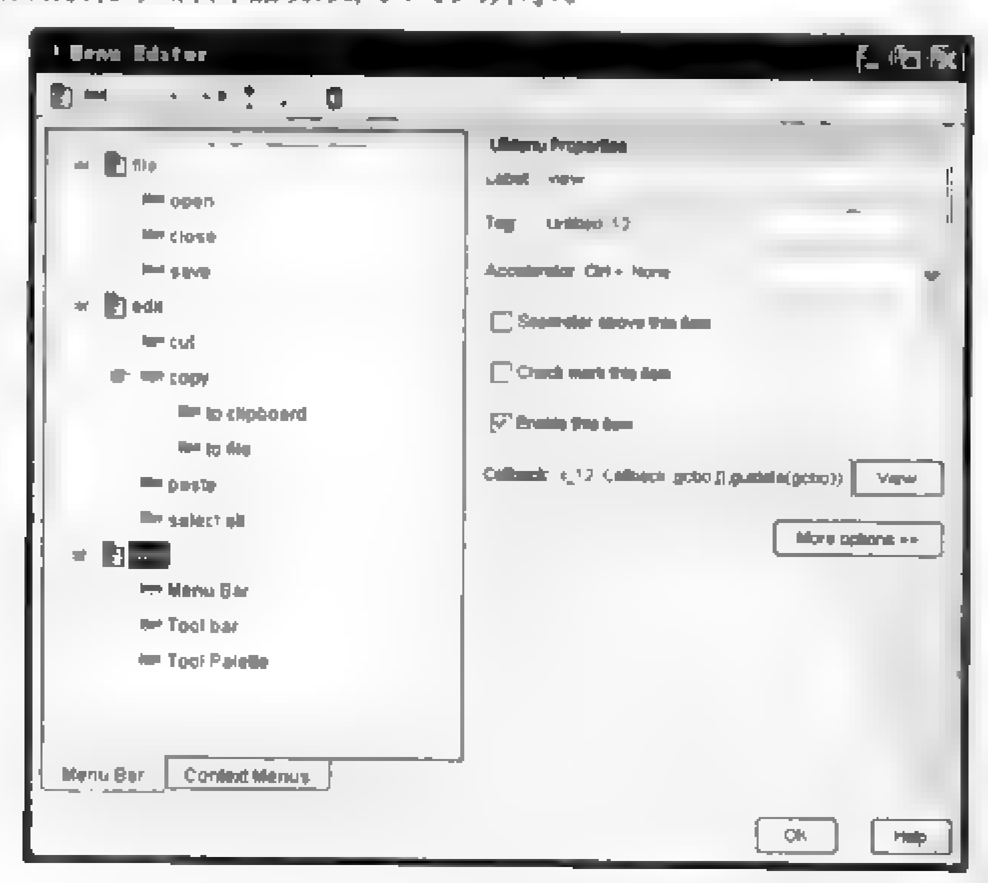


图 14-19　创建完成之后的菜单编辑器

运行该 GUI，所得图形如图 14-20 所示。

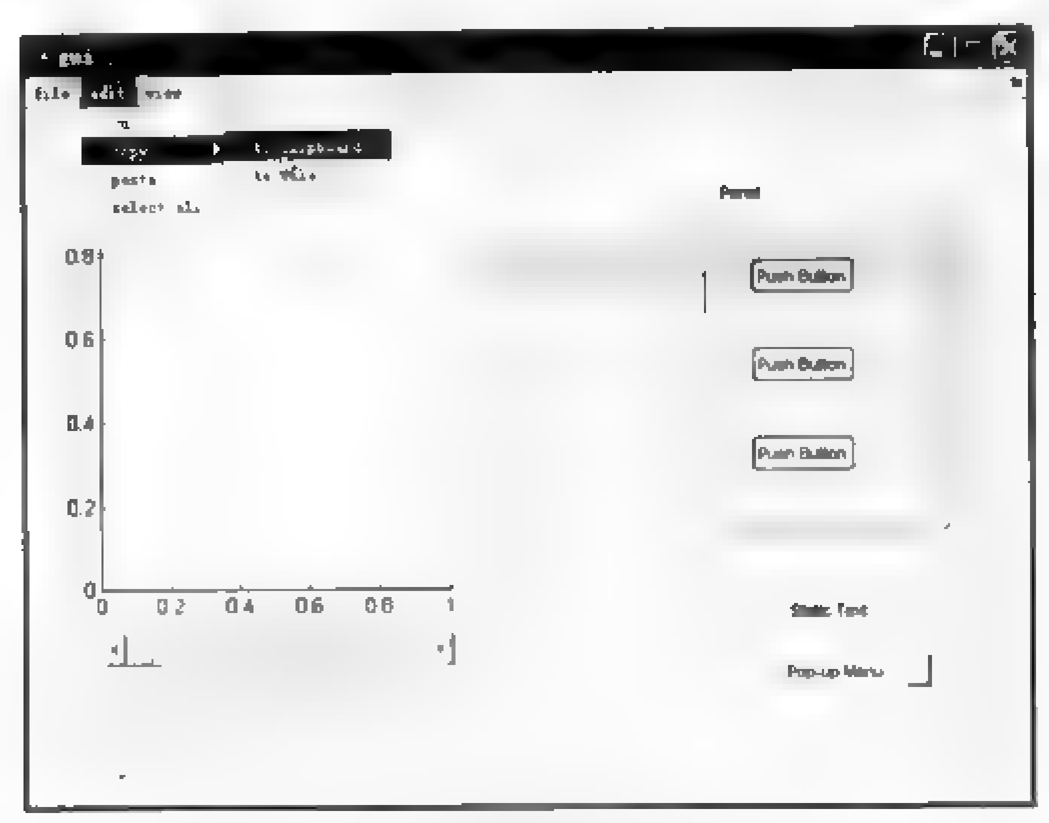


图 14-20　添加菜单之后的 GUI

2. 弹出式菜单的的创建

当用户右击某个对象时，如果该对象被设置了弹出式菜单，那么将有弹出式菜单弹出，用户可以使用 Menu Editor 来定义弹出式菜单并将它们与布局编辑器中的对象相连。

(1) 创建父菜单

所有弹出式菜单中的选项都是一个菜单的子对象，该菜单不在图形菜单栏中显示，欲定义父菜单，可以在 Menu Editor 中单击 New Context Menu 图标，如图 14-21 所示。

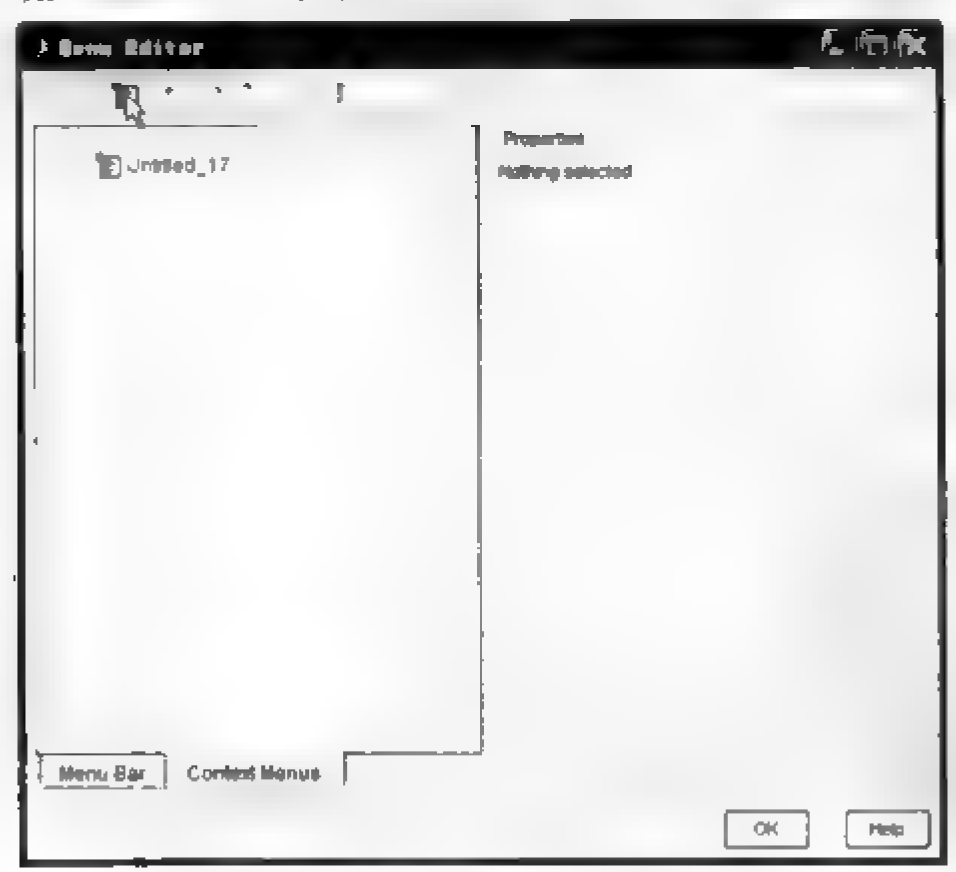


图 14-21　弹出式菜单的的创建

(2) 给弹出式菜单增添菜单项

可以按照给上菜单增添菜单项方法给弹出式菜单增添菜单项，如图 14-22 所示。

图 14-22　给弹出式菜单增添菜单项

(3) 将弹出式菜单与对象相链接

在布局编辑器中，选择你需要定义弹出式菜单的对象，使用 Property Inspector 设置该对象的 UIContextMenu 属性到所需的弹出式菜单，如图 14-23 所示。

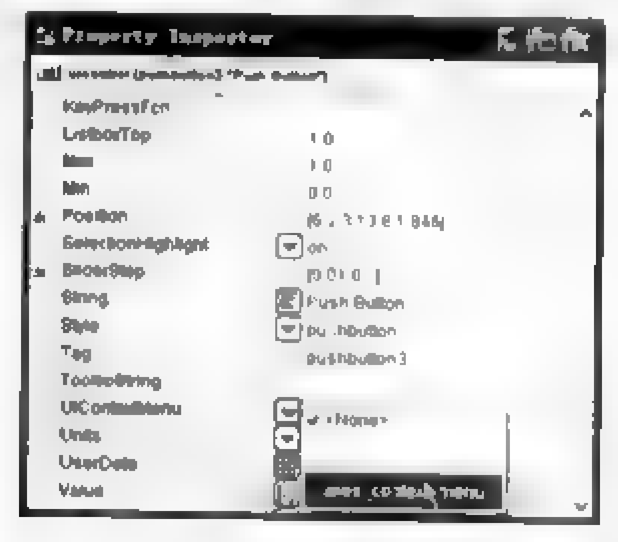


图 14-23　将弹出式菜单与对象相链接

运行 GUI 窗口，所得图形如图 14-24 所示。

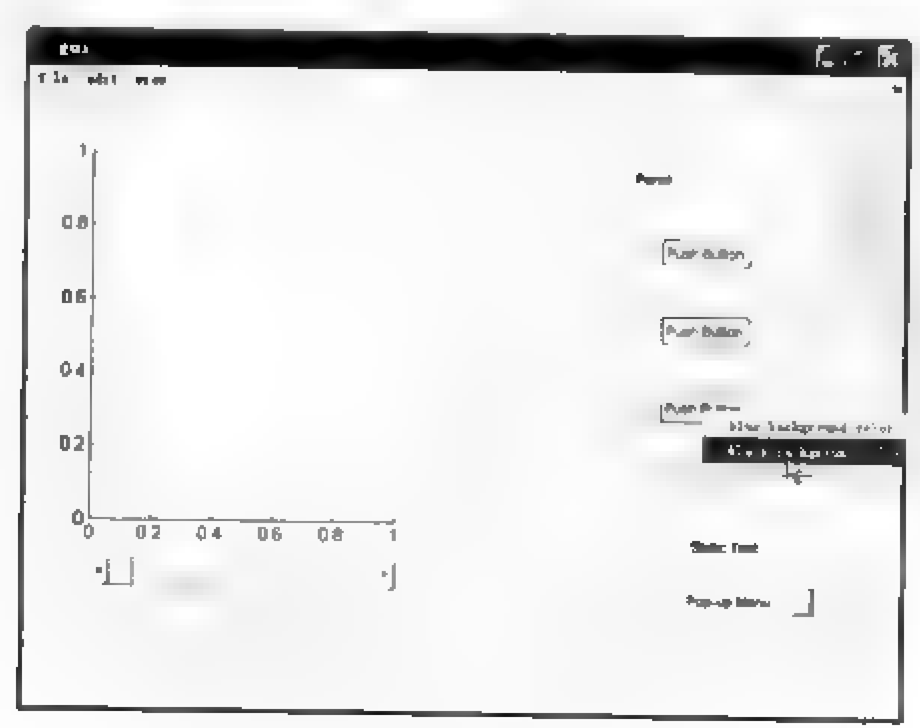


图 14-24　运行弹出式菜单

14.4　GUI 编 程

前边几节已经介绍了 GUI 的基本知识和 GUI 对象的创建及其属性，本节讲述如何进行 GUI 编程，包括 GUI 的 M 文件的理解、GUI 控件的响应的编程以及使用句柄结构进行 GUI 数据操作。通过学习本节，用户可以掌握编制出实用的 GUI 图形用户界面的方法。

14.4.1　GUI 的 M 文件

GUI 的 M 文件由 guide 命令生成，它控制整个 GUI 并决定它对用户的行为(比如单击按钮或选择菜单项)的响应，包含有运行 GUI(包括 GUI 控件的响应)的所有代码。虽然 guide 命令产生了 M 文件的骨架，但是为了实现必要的功能，用户必须对各个响应进行编程，这些是 M 文件的子函数。

1. 与句柄结构共享数据

当运行 GUI 时，M 文件创建一个包含所有 GUI 对象(如控件、菜单和坐标轴)的句柄结构，句柄结构作为一个每个响应的输入来处理。用户使用句柄结构可以实现如下操作。

- 在各响应之间实现数据共享；
- 访问 GUI 数据。

下面对这两种功能分别予以介绍。

(1) 在各响应之间实现数据共享

用户欲取得变量 X 的数据，可以先将句柄结构的一个域设为 X，然后在使用 guidata 函数保存该句柄结构，如下所示：

```
handles.current_data = X;
guidata(hObject,handles)
```

用户可以在其他任何响应中重新得到该变量的值，使用的操作如下。

```
X = handles.current_data;
```

(2) 访问 GUI 数据

用户可以利用句柄结构获取 GUI 控件的任意数据。例如，某个 GUI 有一个弹出式菜单，该菜单的标签是 my_menu，包括 3 个菜单项，这些菜单项的标识字符分别是 chocolate、strawberry 和 vanilla。用户要想使用 GUI 中的另一个控件(比如一个按钮)来在当前所选的菜单项实现某个操作，那么用户可以在该按钮的响应插入如下命令。

```
all_choices = get(handles.my_menu, 'String')
current_choice = all_choices{get(handles.my_menu, 'Value')}
```

上述命令将 current_choice 的值设为 chocolate、strawberry 或 vanilla，具体是哪个值取

决于当前所选的是菜单中的哪个值。

用户可以通过句柄结构访问整个 GUI 的数据，如果该图形的标签是 figure1 那么 handles.figure1 包含了该图形的句柄，例如可以通过如下命令关闭 GUI。

```
delete(handles.figure1)
```

2. M 文件中的函数和响应

用户可以给 GUI 的 M 文件的如下部分增加程序代码。

- 打开函数(Opening function)，该函数在 GUI 可见之前实施操作。
- 输出函数(Output function)，在必要的时候向命令行输出数据。
- 响应(Callbacks)，在用户激活 GUI 中的相应控件时实施操作。

M 文件的常用的输入参数如下。

M 文件中的所有函数都有如下的输入参数与句柄结构相对应。

- hObject，图形或是响应对象的句柄。
- handles，具有句柄或是用户数据的结构。

句柄结构往往在函数的最后阶段进行保存，使用如下命令。

```
guidata(hObject, handles);
```

下面在详细介绍打开函数、输出函数和响应的内容。

(1) 打开函数

打开函数包含有在 GUI 可见之前进行操作的代码，用户可以在打开函数中访问 GUI 的所有控件，因为所有 DUI 中的对象都在调用打开函数之前就已经创建。 如果用户需要在访问 GUI 之前实现某些操作(如创建数据或图形)，那么可以通过在打开函数中增添代码来加以实现。

对于一个文件名为 my_gui 的 GUI 来说，它的打开函数的定义语句如下。

```
function my_gui_OpeningFcn(hObject, eventdata, handles, varargin)
```

在上面的程序语句中，除了上文提到 hObject 和 handles 外，打开函数中还有如下输入参数：

- eventdata，该参数留作以后更高版本的 MATLAB 7.0 所用。
- varargin，参见第 10 章 10.3.1 节。

所有的命令流语句都通过 varargin 传递给打开函数。如果用户调用具有属性名(属性值)的 GUI，那么该 GUI 将按照设定的属性值打开。例如：*my_gui('Position'*, [71.8 44.9 74.8 19.7]) 将在坐标方位为[71.8 44.9]处打开 GUI，该 GUI 的长度为 74.8，宽度为 19.7。

(2) 输出函数

输出函数将输出结果返回给命令行。这一点在用户需要将某个变量传递给另一个 GUI 时尤其实用。

GUIDE 在输出函数中生成如下代码。

```
% --- Outputs from this function are returned to the command line.
function varargout = my_gui_OutputFcn(hObject, eventdata,
handles)
% Get default command line output from handles structure
varargout{1} = handles.output;
```

如果 M 文件中没有 uiwait 命令，那么输出结果仅仅是 GUI 的句柄，该句柄在打开函数中被赋给 handles.output。

要想使 GUI 返回一个不同的输出结果，例如，如果用户要返回一个用户定义的响应(单击一个按钮)可以进行如下操作。

- 在打开函数中增添 uiwait 命令，使得 M 文件在用户激活 GUI 中的某个控件之前处于停止运行状态；
- 对用户期望有回应的 GUI 控件，使得响应能够更新 handles.output 的值，并实施 uiresume 操作。

例如，如果某个 GUI 包含有一个按钮，该按钮的标识字符属性是 Yes，那么在给它的响应增添如下代码后，单击该按钮，GUI 将返回 Yes。

```
handles.output = 'Yes';
guidata(hObject, handles);
uiresume.
```

如果使用 OUT = my_gui 命令调用 GUI ，那么当用户单击 Yes 按钮时，GUI 将返回 OUT = 'Yes' 给命令行。

输出量 varargout 是一个单元型数组，该数组可以包含任意数量的输出参数。默认情况下，GUIDE 只产生一个输出参数 handles.output，如果用户需要创建另外的输出参数，可以在输出函数中增添如下命令。

```
varargout{2} = handles.second_output;
```

用户也可以使用 guidata 命令在任意的响应中设置 handles.second_output 的值。

(3) 响应

当用户激活某个 GUI 控件时，GUI 就对相应的响应实施操作，响应的命令由该控件的标签属性决定。例如：标签为 print_button 的按钮实施如下响应。

```
function print_button_Callback(hObject, eventdata, handles)
```

14.4.2　给 GUI 的控件响应编制程序

本节主要讲述如何给一些特定的 GUI 控件编制程序，这些控件的响应属性描述了一些不同种类的属性。下面对它们分别予以介绍。

1. “开关”按钮(Toggle Button)的响应

在图形界面中添加该控件之后，该控件用于对鼠标的按下与释放做出反应，并由 Value 属性返回，同时可以由 Max 属性和 Min 属性控制 Value 值对事件的响应。默认的情况下，Max 属性值(对应按钮被按下的事件发生时 Value 的值)为 1，而 Min 属性值(对应按钮被释放时 Value 的值)为 0。

在 GUI 的 M 文件中使用如下形式的代码来编制“开关”按钮的响应程序。

```
function togglebutton1_Callback(hObject, eventdata, handles)
button_state = get(hObject,'Value');
if button_state == get(hObject,'Max')
    % toggle button is pressed
elseif button_state == get(hObject,'Min')
    % toggle button is not pressed
end
```

2. 按钮(Radio Buttons)的响应

在图形界面中添加该控件之后，该控件将在指定位置添加按钮，按钮的标识字符由属性 String 控制，而返回值由 Value 值控制。

在 GUI 的 M 文件中使用如下形式的代码来编制 Radio 按钮的响应程序。

```
if (get(hObject,'Value') == get(hObject,'Max'))
    % then rsdio button is selected-take approriate action
else
    % radio button is not selected-take approriate action
end
```

3. “复选框”控件(Check Boxes)的响应

在图形界面中添加该控件之后，该控件将提供复选功能，将显示文本字符串及选择框，当选中时，相应的属性值 Value 为 1，表面对象已经被选中，而在属性 String 中可以定义复选框的文本字符串，通过该控件可以实现多重选择。

在 GUI 的 M 文件中使用如下形式的代码来编制“复选框”控件的响应程序。

```
function checkbox1_Callback(hObject, eventdata, handles)
if (get(hObject,'Value') == get(hObject,'Max'))
    % then checkbox is checked-take approriate action
else
    % checkbox is not checked-take approriate action
end
```

4. “文本框”控件(Edit Text)的响应

在图形界面中添加该控件之后，该控件的属性相当于其他语言设计中的文本框属性，允许用户动态地编辑或是输入文本字符串。

如果需要获取用户在文本框中输入的字符串，可以在响应程序中输入如下代码。

```
function edittext1_Callback(hObject, eventdata, handles)
user_string = get(hObject,'string');
% proceed with callback.
```

此外，用户也可以从文本框控件中获取数值型数据。由于 MATLAB 7.0 从文本框控件中返回的是字符型数据，如果用户需要输入一个数值型的数据，那么，有必要将字符型数据转换为数值型数据，可以通过使用 str2double 命令实现以上操作，如果用户输入的是非数值型字符串，那么该命令将返回 NaN。

用户可以在文本框控件的响应程序中使用如下代码，该程序段可以将输入文本框中的字符型数据转换为双精度类型的数据，并检查该双精度类型的数据是否为 NaN。

```
function edittext1_Callback(hObject, eventdata, handles)
user_entry = str2double(get(hObject,'string'));
if isnan(user_entry)
    errordlg('You must enter a numeric value','Bad Input','modal')
end
% proceed with callback..
```

5. “滚动条”控件(Sliders)的响应

该控件在图形界面中显示 Windows 操作系统中常见的滚动条，该控件最重要的属性是 SliderSkip，该属性的属性值为一个仅有两个元素的数组，第 1 个元素表示单击滚动条两侧的箭头按钮时，滑块向箭头方向的移动距离，默认值为滚动条的 1%；第 2 个表示单击滚动条时，滑块向单击点移动的距离，默认值为滚动条的 10%。

用户可以通过如下代码来获取“滚动条”控件的当前值：

```
function slider1_Callback(hObject, eventdata, handles)
slider_value = get(hObject,'Value');
% proceed with callback...
```

6. “列表框”控件(List Boxes)的响应

该控件列出选项列表，并通过列表选择一个或多个选项，选项数目由属性 Max 和 Min 控制，选项的返回值将由属性 Value 带出，Value 值为 1 时，表示第 1 个选项，为 2 时表示第 2 个选项，依此类推。

当鼠标被释放或是采用如下键盘输入时，MATLAB 7.0 将对列表框的响应做出评估。

- 方向键(arrow keys)改变 Value 属性并引发响应实施。
- Enter 键或是空格键不改变 Value 属性但是引发响应实施。

7. “弹出菜单”控件(Pop-Up Menus)的响应

该控件将提供互斥的一组选项列表供用户选择，在弹出菜单的右侧有一向下的箭头，单击后显示所有的对象列表，属性 String 用于接受对象名，属性 Value 将返回选中的对象

在对象列表中的下标。

在下边的代码中，首先检查所选项的值，并在这个值的基础上使用 switch 语句来实施操作。

```
function popupmenu1_Callback(hObject, eventdata, handles)
val = get(hObject,'Value');
switch val
case 1
% The user selected the first item
case 2
% The user selected the second item
% proceed with callback...
```

在下边的程序中，响应获取弹出式菜单中的实际的字符串，这种途径适宜于程序需要给弹出式菜单动态加载内容的情况。需要注意的是需要将由 String 属性返回的值由单元型数组转换为字符串。

```
function popupmenu1_Callback(hObject, eventdata, handles)
val = get(hObject,'Value');
string_list = get(hObject,'String');
selected_string = string_list{val}; % convert from cell array
                                    % to string
% proceed with cal
```

8. 控件板(Panels)的响应

控件板可以将相关的控件编为一个组，控件板中可以包括各种控件，坐标轴对象以及交互式控件，控件板中每一个控件的位置都是与控件板的相对位置。如果控件板的尺寸发生了改变，需要重新对其中的控件进行定位。

下边的代码将取得控件板 uipanel2 的位置属性，一旦取得了位置数据，用户就可以利用它来改变子控件的位置。

```
function uipanel2_ResizeFcn(hObject, eventdata, handles)
% hObject     handle to uipanel1 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB 7 0
% handles     structure with handles and user data (see GUIDATA)
pos = get(hObject,'Position');
% proceed with callback...
```

此外，GUI 控件还包括“坐标轴”控件(Axes)、“静态文本”控件(Static Text)和“选择”控件(Select)等，关于它们的详细信息用户可以查看 MATLAB 的帮助系统，在此不再赘述。

14.4.3　使用句柄结构进行 GUI 数据操作

GUIDE 提供了一种机制来存储和检索共享的数据，这些操作都通过包含有 GUI 控件句柄的相同结构来实现，这种机制就称为句柄结构。句柄结构包含有 GUI 的所有控件的句柄，它在 M 文件中被传递给每一个响应。因此，该响应对保存任意共享数据十分有用。

例 14-1　在响应之间传递数据。

本例演示了如何利用句柄结构在响应之间传递数据，使用以下的方式在在一个“滚动条”控件和“文本框”控件之间传递数据。

- 用户移动滚动条时，文本框中显示滚动条的当前值。
- 用户在文本框中输入一个值，滚动条自动移动到与该值对应的位置。
- 如果用户在文本框中输入的值超出了滚动条值的范围，即该值不在 0 和 1 之间。程序在文本框中输出输入错误值的次数。

解：进行如下操作。

(1) 首先在命令窗口中输入 duide，并按 Enter 键确认，然后打开一个空白的 GUI 模块，布置控件如图 14-25 所示。

图 14-25　控件的布置

(2) 在打开函数中定义如下数据域。

保存 GUI，系统将自动调出对应的 M 文件。该 GUI 记录了用户在“文本框”控件中输入错误值的次数并将该数字存储在句柄结构的一个域中，用户可以在打开函数中定义该域，比如，可以使用如下语句将其定义为 number errors：

```
handles.number_errors = 0;
```

将上述命令行键入到下面由 GUIDE 自动嵌入打开函数的命令行之前。

```
guidata(hObject, handles); % Save the updated structure
```

guidata 命令将句柄结构保存，这样就可以在响应中调用该句柄结构。

应当注意的是，用户欲保存对句柄结构所做的修改，必须在执行这些改变的代码后加上命令 guidata(hObject, handles)。

(3) 根据“滚动条”控件的响应设置“文本框”控件的值。

在滚动条控件的响应函数中可以设置如下命令，当用户用鼠标移动该滚动条并释放鼠标后，该命令可以根据滚动条的位置对文本框中的值进行更新。

```
set(handles.edit1,'String',...
      num2str(get(handles.slider1,'Value')));
```

该命令行包含如下 3 条命令语句：

- get 命令获取滚动条的当前值。The get command obtains the current value of the slider.
- num2str 命令将该值转换为字符型变量。
- set 命令将文本框控件的 String 属性重设为当前值。

(4) 根据“文本框”控件的响应设置“滚动条”控件的值。

“文本框”控件的响应将“滚动条”控件的值设置为用户在文本框中键入的值，在检查该值是否在 0 和 1 之间后，如果该值超出了范围，文本框中显示一条信息，提示用户已经输入了多少次错误的数据。命令行代码如下。

```
val = str2double(get(handles.edit1,'String'));
% Determine whether val is a number between 0 and 1
if isnumeric(val) & length(val)==1 & ...
      val >= get(handles.slider1,'Min') & ...
      val <= get(handles.slider1,'Max')
      set(handles.slider1,'Value',val);
else
% Increment the error count, and display it
      handles.number_errors = handles.number_errors+1;
      guidata(hObject,handles); % store the changes
      set(handles.edit1,'String',...
      ['You have entered an invalid entry ',...
num2str(handles.number_errors),' times.']);
end
```

(5) 运行 GUI。

设置为代码之后，运行该 GUI，如图 14-26 所示。

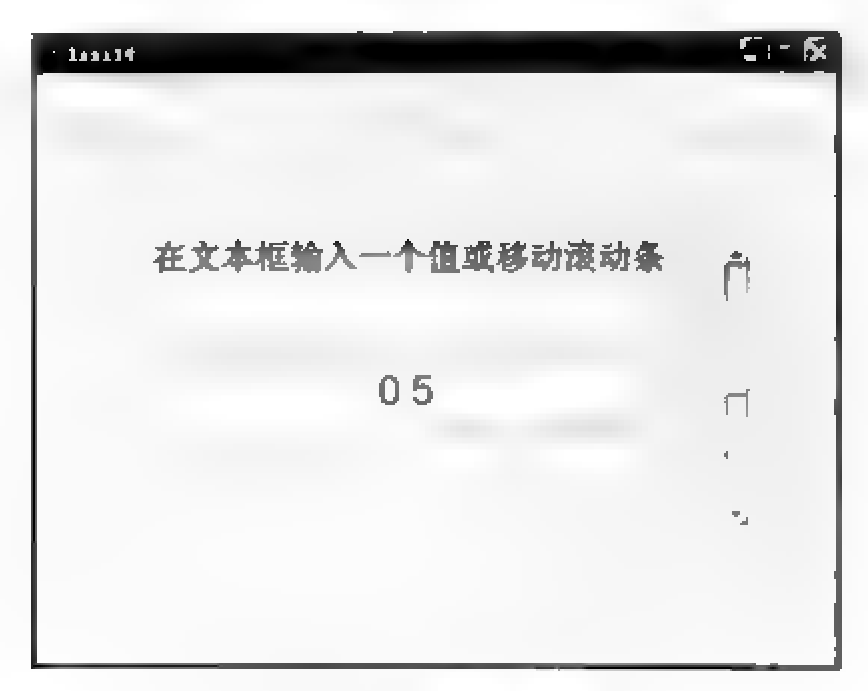


图 14-26　运行 GUI

运行之后，如果用户在文本框中输入在 0 到 1 之间的某个数，并在文本框外边单击，滚动条自动移动到与该值对应的位置上；或者是将滚动条移动到某个位置，释放鼠标，文本框中出现与滚动条位置对应的值。

如果输入的数字不在该范围内，将在文本框中出现如下的信息，如图 14-27 所示。

图 14-27　提示错误信息

14.5　习　　题

1. 简述在 MATLAB 7.0 中创建 GUI 的步骤。
2. 简述 GUI 控件的种类，它们各自的功能是什么？
3. 什么是响应函数？它在 GUI 中的作用是什么？响应函数是如何工作的？
4. 简述句柄结构的用途。
5. 创建一个 GUI，使用一个弹出式控件来选择 GUI 的底色。

6. 创建一个 GUI，绘制曲线 $y(x)=ax^2+bx+c$，该 GUI 将提示用户输入 a、b 和 c 的值，并输入 x 的取值范围。

7. 给习题 14.4 所创建的 GUI 增添一个菜单，该菜单包括两个子菜单，可以选择线型和颜色。

第15章　微分和积分

微积分是大学数学的重要组成部分，几乎是每一个理工科学生所必修的课程，它是各个学科的基础，科学研究和工程实践中都有着广泛的应用。在 MATLAB 7.0 语言中，提供了许多求解微积分的函数，同时，用户也可以自己编写函数来求解复杂的问题。

15.1　数 值 微 分

在工程实际、科学研究和日常学习中，除了进行数值拟合和插值外，有时候还需要根据已知的数据点，求某些点的一阶或高阶导数，此时，就需要使用数值微分。MATLAB 7.0 提供了 3 个功能函数，它们是 diff 函数求数值微分、gradient 函数求近似梯度以及 jacobian 函数求多元函数的导数。应用这 3 个功能函数几乎可以解决所有的微分问题。

15.1.1　使用 diff 函数求数值微分

在 MATLAB 7.0 中，使用 diff 函数可以求解数值微分。其使用格式如下。

- $diff(x)$ 命令求向量 x 的微分，所得值为[$x(2)-x(1)$　$x(3)-x(2)$ … $x(n)-x(n-1)$]。
- $diff(x)$ 命令求矩阵 x 的微分，所得值为矩阵的差分[$x(2)-x(1)$　$x(3)-x(2)$ … $x(n)-x(n-1)$]。
- $diff(x)$ 命令对 n 维数组 x，所得值为沿第一个相关维的差分值。
- $diff(x,n)$ 命令用来求 n 阶差分值，若 $n > size(x, DIM)$，diff 函数将先计算可能的连续差分值，直到 $size(x, DIM) = 1$。然后 diff 函数沿任意 N+1 维进行差分计算。
- $diff(x,n,DIM)$ 命令用来求 n 阶的差分值，如果 $n > size(x, DIM)$，函数将返回空数组。

例 15-1　使用 diff 函数求数值微分。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> h = .001;
>> x = 0:h:pi;
>> diff(sin(x.^2))/h;          %is an approximation to 2*cos(x.^2).*x
>> diff((1:10).^2)
ans =
     3     5     7     9    11    13    15    17    19
```

```
>> X = [3 7 5
        0 9 2]
X =
     3     7     5
     0     9     2
>> diff(X,1,1)
ans =
    -3     2    -3
>> diff(X,1,2)
ans =
     4    -2
     9    -7
>> diff(X,2,2)        %is the 2nd order difference along the dimension 2
ans =
    -6
   -16
>> diff(X,3,2)        %is the empty matrix.
ans =
   Empty matrix: 2-by-0
>>
```

15.1.2　使用 gradient 函数求近似梯度

在 MATLAB 7.0 语言中，使用 gradient 函数求解近似梯度。其使用格式如下。

- $[fx, fy] = gradient(f)$ 命令返回矩阵 f 的数值梯度，fx 相当于 $\frac{df}{dx}$，即在 x 方向(列)的差分值。fy 相当于 $\frac{df}{dy}$，即在 y(列)方向的差分值。各个方向的间隔设为 1。当 f 是一个向量时，df = *gradient(f)*命令返回一个一维向量。
- $[fx, fy] = gradient(f, h)$ 命令使用 h 作为各个方向的间隔点，这里 h 为一个数量。
- $[fx, fy] = gradient(f, hx, hy)$ 命令使用 hx 和 hy 为指定间距，其中 f 为二维函数。而 hx 和 hy 可以为向量或数量，但 hx 和 hy 为向量时，它们的维数必须和 f 的维数相匹配。
- $[fx, fy, fz] = gradient(f)$ 命令返回一个 f 的三维梯度，其中 f 是一个三维向量，fz 相当于 $\frac{df}{dz}$，即在 z 方向的差分。$gradient(f, h)$ 命令使用 h 作为各个方向的间距，其中 h 为一个数量。
- $[fx, fy, fz] = gradient(f, hx, hy, hz)$ 命令使用 hx 、 hy 和 hz 为指定间距。
- $[fx, fy, fz] = gradient(f, ...)$ 命令在 f 为 N 维数组时作相似扩展。

例 15-2　使用 gradient 函数求近似梯度。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> [x,y] = meshgrid(-2:.2:2, -2:.2:2);
>> z = x .* exp(-x.^2 - y.^2);
>> [px,py] = gradient(z,.2,.2);
>> contour(z)
>> hold on
>> quiver(px,py)
>>
```

运行结果如图 15-1 所示。

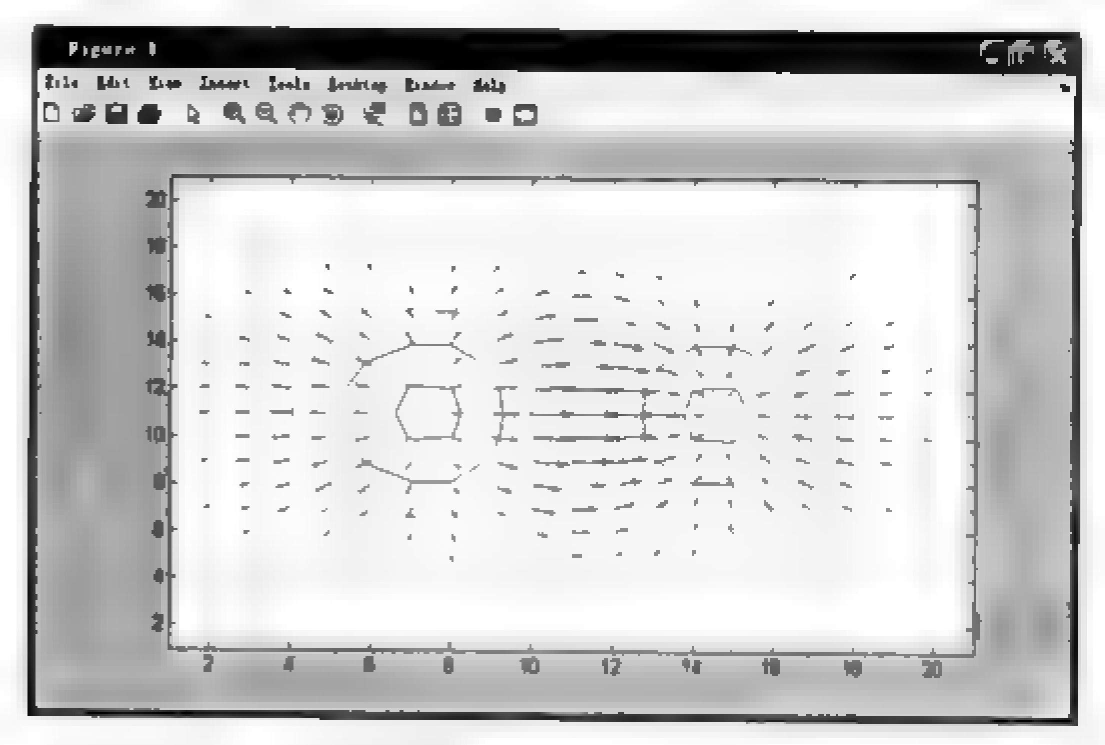


图 15-1　gradient 函数求近似梯度

15.1.3　jacobian 函数求多元函数的导数

在 MATLAB 7.0 语言中，使用 jacobian 函数求解多元函数的导数，其使用格式如下。

- *jacobian*(f,v) 命令计算向量 f 对向量 v 的 Jacobian 矩阵，所得结果的第 i 行、第 j 列的值为 $\mathrm{d}f(i)/\mathrm{d}v(j)$。当 f 为数量时，所得值为 f 的梯度。v 也可以为数量，不过此时该命令仅相当于 $diff(f,v)$。

例 15-3　使用 jacobian 函数求多元函数的导数。

```
>> syms x y z
>> jacobian([x*y*z; y; x+z],[x y z])
ans =
[ y*z, x*z, x*y]
[    0,    1,    0]
[    1,    0,    1]
>> syms u v
>> jacobian(u*exp(v),[u;v])
ans =
```

```
[    exp(v), u*exp(v)]
>>
```

15.2　函数的数值积分

15.2.1　一元函数的数值积分

在科学研究和工程实践中，除了要进行微分外，有时候还要求曲线下的面积，此时，就要用到积分方面的知识。从理论上来说，可以利用牛顿－莱不尼兹公式来求已知函数的积分，但是，工程实际中的函数往往十分复杂，采用理论的方法很难找到原函数。在 MATLAB 7.0 语言中，提供了一些用于采用数值方法求解积分的函数，使用它们，用户可以得到满意的精度。主要有 cumsum、trapz、quad 和 quad8 等，本节将对它们的使用方法予以介绍。

1. 矩形求积

在 MATLAB 7.0 语言中，采用矩形求积法求解积分由 cumsum 函数实现。其使用格式如下。

- 对向量 x，$cumsum(x)$ 命令返回一个向量，该向量的第 N 个元素是 x 的前 N 个元素的和。
- 对矩阵 x，$cumsum(x)$ 命令返回一个和 x 同型的矩阵，该矩阵的列即为对 x 的每一列的累积和。
- 对 N 维数组 x，$cumsum(x)$ 命令从第一个非独立维开始操作。
- $cumsum(x, DIM)$ 命令中，参数 DIM 指明是从第一个非独立维开始的。

例 15-4　cumsum 函数采用求积法来求解积分。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> x1=[1 2 3 4 5 6 7 8 9]
x1 =
     1     2     3     4     5     6     7     8     9
>> cumsum(x1)
ans =
     1     3     6    10    15    21    28    36    45
>> x2=[1 2 3 ;4 5 6,7 8 9]
x2 =
     1     2     3
     4     5     6
     7     8     9
>> cumsum(x2)
ans =
```

```
         1        2        3
         5        7        9
        12       15       18
>> cumsum(x2,1)
ans =
         1        2        3
         5        7        9
        12       15       18
>> cumsum(x2,2)
ans =
         1        3        6
         4        9       15
         7       15       24
>> cumsum(x2,3)
ans =
         1        2        3
         4        5        6
         7        8        9
>>
```

例 15-5　使用 cumsum 函数求函数 sin(x)在[0,10]区间内的积分。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> x=0:0.1:10;
>> y=sin(x);
>> z=cumsum(y)*0.1;
>> plot(x,y,'r-',x,z,'k*')
>>
```

运行结果如图 15-2 所示。

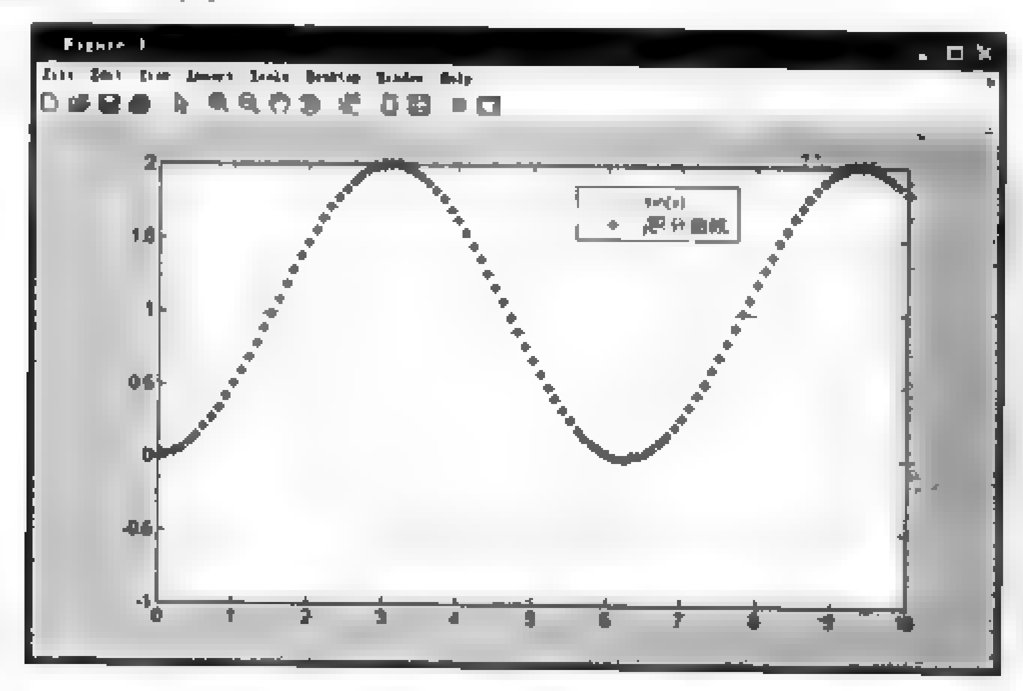


图 15-2　使用 cumsum 函数求 sin(x)的积分

从图 15-2 可以看出，求出的积分曲线和余弦曲线形状相同，这与理论计算的结果是相

符合的。

2. trapz 函数(梯形求积)

在 MATLAB 7.0 语言中，采用梯形求积法求解积分由 trapz 函数实现。其使用格式如下。

- $z = trapz(y)$ 命令采用梯形法近似求解 y 的积分近似值。当间距不是 1 时，采用间距乘以 z 的方法来求解。对向量 y，$trapz(y)$ 命令返回 y 的积分；对矩阵 y，$trapz(y)$ 命令返回一个行向量，该行向量的元素值为矩阵 y 对应列向量的积分值；对 N 维列向量，$trapz(y)$ 命令从第一个非独立维开始计算。
- $z = trapz(x, y)$ 命令使用梯形法求解 y 对 x 的积分值。其中 x 和 y 必须是具有相同长度的向量，或是 x 是一个列向量，而 y 的第一个非独立列的维数是 $length(x)$，本函数将沿此维开始操作。
- $z = trapz(x, y, DIM)$ 命令或是 $z = trapz(y, DIM)$ 命令求 y 的交叉维 DIM 的积分，其中 x 的长度必须等于 $size(y, DIM)$。

例 15-6　trapz 函数采用梯形求积法来求解积分。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> X=[1 2 3 4 5 6 7 8 9 10]
X =
     1     2     3     4     5     6     7     8     9    10
>> Z1=trapz(X)
Z1 =
   49.5000
>> Y = [0 1 2;3 4 5;6 7 8]
Y =
     0     1     2
     3     4     5
     6     7     8
>> trapz(Y,1)
ans =
     6     8    10
>> trapz(Y,2)
ans =
     2
     ▮
    15
>> x=[1 2 3]
x =
     1     2     3
>> trapz(x,Y)
ans =
     6     8    10
>>
```

从上边的两个例子可以看出，cumsum 函数对向量求积分时，返回一个向量，对矩阵求积分时，返回一个矩阵，而 trapz 函数对向量求积分时，返回一个数值，而对矩阵求积分时，返回一个向量，这是它们之间不同之处。

3. 自适应法(Simpson 法)

在 MATLAB 7.0 语言中，使用自适应法(Simpson 法)进行积分由 quad 函数来实现。其使用格式如下。

- $q = quad(fun,a,b)$ 命令使用 Simpson 法则的自适应法求函数 f 从 a 到 b 的相对误差为 1.e-6 的积分近似值，其中函数 $y = fun(x)$ 必须接受向量 x 并返回一个向量 f。
- $q = quad(fun,a,b,tol)$ 命令使用了一个绝对误差容度 tol 代替默认值 1.e-6。当 tol 的值比较大的时候，可以使计算速度明显加快，但是计算精度会有所降低。quad 函数在 MATLAB 5.3 版本中使用的默认误差值为 1.e-3。
- $[q, fcnt] = quad(...)$ 命令返回函数计算的步数。
- 当 $trace$ 不为 0 时，$quad(fun,a,b,tol,trace)$ 命令返回[fcnt a b-a Q]的值。

例 15-7　quad 函数使用自适应法(Simpson 法)来求解积分。

函数 $f = \int_0^1 \frac{x^2}{(1+\sin(x)+x^2)}\,dx$

先编制 M 文件如下。

```
function f=fun1(x)
f=x.^2./(1+sin(x)+x.^2);
end
```

将该 M 文件以 fun1.m 为函数名保存。继续在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> quad('fun1',0,1)
ans =
     0.2301
>>
```

4. 高阶自适应法(Newton-Cotes 法)

在 MATLAB 7.0 语言中，quad8 函数采用自适应的 Newton-Cotes 法求解积分。其使用格式如下。

- $q = quad8(fun,a,b)$ 命令使用高阶自适应法(Newton-Cotes 法)求解函数 fun 在积分区间[A,B]上相对积分为 1.e-3 的积分近似值。函数 fun 接受一个向量 x 并返回向量 y。当积分超出递归限时，$q = Inf$。
- $q = quad8(fun,a,b,tol)$ 命令使用高阶自适应法(Newton-Cotes 法)求解函数 fun 在积分区间[A,B]上的积分近似值。其中向量 tol 可以包含两个元素，即 $OL = [rel\ \ tol$

abs_tol]，其中元素 *rel_tol* 为相对误差，元素 *abs_tol* 为绝对误差。

- $q = quad8(fun,a,b,tol,trace)$ 命令中，当 *trace* 不为 0 时，将绘制出由点组成的图形。

例 15-8　quad8 函数使用自适应法(Simpson 法)来求解积分。

函数 $f = \int_0^3 \frac{x^2}{e^{-x}}dx$

先编写 M 文件如下。

```
function f=fun2(x)
f=x ^2./exp(-x);
end
```

在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> quad8('fun2',0,3)
ans =
    98.4277
>> a=quad8('fun2',1,3,[1e-10,1e-11])
a =
      97.7094
>> vpa(a,10)
ans =
97.70940279
>> quad8('fun2',1,3,1e-6,1)
        18        1.0000000000        1.00000000e+000        97.7094029629
ans =
    97.7094
>>
```

15.2.2　二元及三元函数的数值积分

15.2.1 节介绍了一元函数的数值积分知识，本节将介绍一些关于二元函数的积分知识，在 MATLAB 7.0 语言中，使用 dblquad 函数和 quad2dggen 函数来求解二元函数的积分。

1. dblquad 函数求任意区域的积分

在 MATLAB 7.0 语言中，使用 dblquad 函数求矩形区域的积分。其使用格式如下。

- $dblquad(fun,XMIN,XMAX,YMIN,YMAX)$ 命令调用函数 quad 在矩形区域 $[XMIN,XMAX,YMIN,YMAX]$ 上计算二元函数 $fun(x,y)$ 的二重积分。输入向量 x，标量 y，则 $fun(x,y)$ 必须返回一个用于积分的向量。
- $dblquad(fun,XMIN,XMAX,YMIN,YMAX,tol)$ 命令用指定的精度 *tol* 代替默认的精度 10^{-6}，再进行计算。

- *dblquad(fun,XMIN,XMAX,YMIN,YMAX,tol,@quadl)* 命令用指定的算法 *quadl* 代替默认算法 quad。*quadl* 的取值由 *@quadl* 指定且与命令 quad 与 *quadl* 有相同调用次序的函数句柄。
- *dblquad(fun,XMIN,XMAX,YMIN,YMAX,tol,@quadl,p1,p2...)* 命令将可选参数 *p1*，*p2*，…等传递给函数 *fun(x,y,p1,p2,...)*。若 *tol* 和 *quadl* 都是空矩阵时，则使用默认精度和算法 quad。

例 15-9　使用 dblquad 函数在矩形区域求二重积分。

函数 $f = \dfrac{y}{\sin(x)} + x * e^y$

```
>> Q = dblquad(inline('y*sin(x)+x*cos(y)'), pi, 2*pi, 0, pi)
Q =
    -9.8696
>>
>> Q = dblquad(inline('y*sin(x)+x*cos(y)'), pi, 2*pi, 0, pi,1e-10)
Q =
    -9.8696
>> vpa(Q,10)
ans =
-9.869604401
>>
>> Q = dblquad(@integrnd1, pi, 2*pi, 0, pi)
Q =
    -9.8696
>>
```

其中 integrnd1 函数的 M 文件如下：

```
function z = integrnd1(x, y)
                z = y*sin(x)+x*cos(y);
end
```

例 15-10　使用 dblquad 函数在非矩形区域求二重积分。

函数 $f = \sqrt{\max(1-(x^2+y^2),0)}$

```
>> dblquad(inline('sqrt(max(1-(x.^2+y.^2),0))'),-1,1,-1,1)
ans =
     2.0944
>>
```

2. triplequad 函数求三元函数的积分

在 MATLAB 7.0 中，使用 triplequad 函数求三元函数的积分。其使用格式如下。

- *triplequad(fun,XMIN,XMAX,YMIN,YMAX,ZMIN,ZMAX)* 命令求函数 *fun*(x,y,z) 在矩形区间 [*XMIN,XMAX,YMIN,YMAX,ZMIN,ZMAX*] 上的积分值。函数 *fun*(x,y,z) 必须接受向量 x 以及标量 y 和 z 并返回一个积分向量。
- *triplequad(fun,XMIN,XMAX,YMIN,YMAX,ZMIN,ZMAX,tol)* 命令使用 *tol* 作为允许的误差值，取代默认值 1.e-6。
- *triplequad(fun,XMIN,XMAX,YMIN,YMAX,ZMIN,ZMAX,tol,@quadl)* 命令使用指定的算法代替默认的算法 quad。
- *triplequad(fun,XMIN,XMAX,YMIN,YMAX,tol,ZMIN,ZMAX,,@myquadf)* 命令使用自定义的算法 *myquadf* 取代默认值 quad。函数 *myquadf* 必须和 quad 与 *quadl* 两函数有相同调用次序的函数句柄。
- *triplequad(fun,XMIN,XMAX,YMIN,YMAX,ZMIN,ZMAX,tol,@quadl,p1,p2,...)* 命令将可选参数 *p*1， *p*2，…等传递给函数 *fun*$(x,y,p1,p2,...)$。若 *tol* 和 *quadl* 都是空矩阵时，则使用默认精度和算法 quad。

例 15-11　使用 triplequad 函数求三元函数的积分。

函数 $f = y*\sin(x)+z*\cos(x)$

```
>>  Q = triplequad(inline('y*sin(x)+z*cos(x)'), 0, pi, 0, 1, -1, 1)
Q =
     2.0000
>>
>> Q = triplequad(inline('y*sin(x)+z*cos(x)'), 0, pi, 0, 1, -1, 1,1e-9)
Q =
     2.0000
>> vpa(Q,10)
ans =
2.000000000
>>
```

可见，ans=2.000 即为本例的精确值。此外，用户还可以使用如下形式求解积分值。

```
>> Q = triplequad(@integrnd, 0, pi, 0, 1, -1, 1)
Q =
     2.0000
>>
```

其中 integrnd 函数的 M 文件如下：

```
function f = integrnd(x, y, z)
                f = y*sin(x)+z*cos(x);
end
```

15.3 习　　题

1. 求函数 $y=\sin(x^2)^3$ 的微分。

2. 求函数 $y=\mathrm{e}^{x+y^2}$ 的近似梯度。

3. 求多元函数 $f(x,y,z)=\begin{bmatrix}\sin(x)+\cos(y)+\sin(z)\\ x^2+\dfrac{y}{z}\\ x^y+\mathrm{e}^z\end{bmatrix}$ 的 Jacobian 矩阵。

4. 计算积分 $\int_{-1}^{1}x+x^3+x^5\mathrm{d}x$ 的值。

5. 计算积分 $\int_{1}^{10}\sin(x)+\cos(x)\mathrm{d}x$ 的值。

6. 计算积分 $\int_{2}^{6}\mathrm{e}^{-\frac{x}{2}}\mathrm{d}x$ 的值。

7. 计算积分 $\int_{1}^{10}\dfrac{x}{x^4+4}\mathrm{d}x$ 的值。

8. 计算积分 $\int_{1}^{10}\int_{1}^{10}\sin(y)\dfrac{x+y}{4+x^2}\mathrm{d}x\mathrm{d}y$ 的值。

9. 计算积分 $\int_{1}^{10}\int_{1}^{y}y\dfrac{x+y}{4}\mathrm{d}x\mathrm{d}y$ 的值。

10. 计算积分 $\int_{0}^{3}z\int_{1}^{10}\int_{1}^{y}y\dfrac{x+y}{4}\mathrm{d}x\mathrm{d}y\mathrm{d}z$ 的值。

第16章 拟合和插值

在实际生活中，完全像数学上那样精确的数值是几乎不存在的，比如统计学中的样本空间的数可能是杂乱无章的，用户可以通过不同的方式测得一些采样点。但是，由于这些采样点的不规则性，必须对它们进行分析和处理。使用插值和拟合方法之后，可以从不规则的事物中找到其内在的规律，从而可以指导实践。

16.1 最小二乘法实现曲线拟合

在实际的工程应用中，人们往往只能测得一些分散的数据点，为了从这些分散的数据点中找到其内在的规律性，就需要利用这些分散的数据点，运用最小二乘法、多项式或其他的已知函数等方法来生成一个新的多项式或是函数来逼近这些已知点。由于 MATLAB 7.0 语言强大的计算功能和绘图功能，使得用户可以很方便地进行曲线拟合并绘制出曲线拟合图。

曲线拟合涉及两个基本问题：最佳拟合意味着什么？应该用什么样的曲线？由于可用许多不同的方法定义最佳拟合，并存在无穷数目的曲线。而当最佳拟合被解释为在数据点的最小误差平方和，且所用的曲线限定为多项式时，那么曲线拟合是相当简捷的。数学上，称为多项式的最小二乘曲线拟合，如图 16-1 所示。虚线和标志的数据点之间的垂直距离是在该点的误差。对各数据点距离求平方，并把平方距离全加起来，就是误差平方和。这条虚线是使误差平方和尽可能小的曲线，即是最佳拟合。最小二乘这个术语是使误差平方和最小的省略说法。

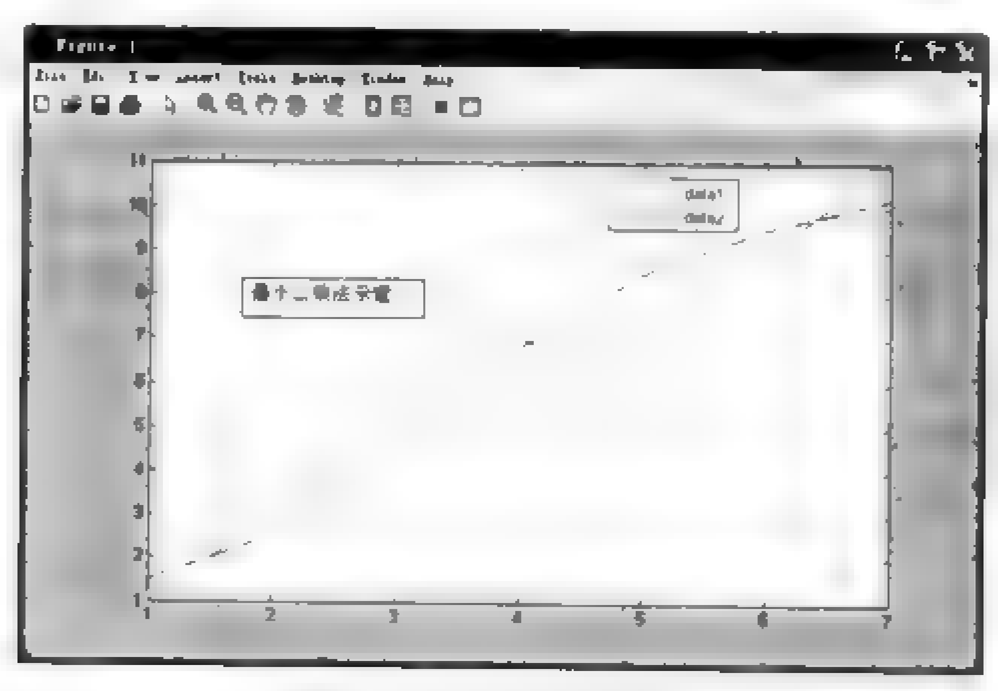


图 16-1 最小二乘拟合曲线

在 MATLAB 7.0 语言中，使用 polyfit 函数来求解最小二乘曲线拟合问题。其使用方法如下。

- $polyfit(x,y,n)$ 命令用最小二乘法对所给数据进行 n 阶多项式拟合，返回拟合多项式 $p(x)$，使得 $p(x(i)) \sim= y(i)$。
- $[p,s] = polyfit(x,y,n)$ 命令不仅返回多项式的系数 p，还返回用函数 polyval 获得的误差分析报告。如果数据中的错误数据 y 服从独立正态分布，那么 polyval 函数对错误数据的预报率为 50%。
- $[p,s,mu] = polyfit(x,y,n)$ 命令返回拟合多项式的系数 $xhat = (x - mu(1)/mu(2))$，这里 $mu(1) = mean(x)$，$mu(2) = std(x)$。
- 当 $n >= length(x)$ 时，或是 x 被重复赋值，系统将产生出错信息。

例 16-1　使用 polyfit 函数来求解最小二乘曲线拟合。

假设现在有如表 16-1 所列试验数据，要求其最小二乘拟合曲线。

表 16-1　最小二乘拟合曲线数据

X	1	2	3	4	5	6	7	8	9	10
Y	1	3	11	12	28	32	45	70	80	104

解：编写程序如下。

```
>>x=[1 2 3 4 5 6 7 8 9 10];
>>y=[1 3 11 12 28 32 45 70 80 104];
```

为了用 polyfit 函数，用户必须给函数赋予上面的数据和用户希望最佳拟合数据的多项式的阶数。如果用户选择 N=1 作为阶数，就会得到最简单的线性近似，通常称为线性回归。编制程序如下。

```
>> a=polyfit(x,y,1)
a =
     9.8000      9.2000
>>
```

为了查看拟合的多项式与原来数据的拟合程度，用户可以绘制 MATLAB 7.0 曲线来加以验证。继续在命令窗口中输入如下命令，并按 Enter 键确认，如图 16-2 所示。

```
>>x1=1:0.05:10;
>>y1=a(1)*x1+a(2);
>>plot(x,y,'*',x1,y1,'-r')
>>
```

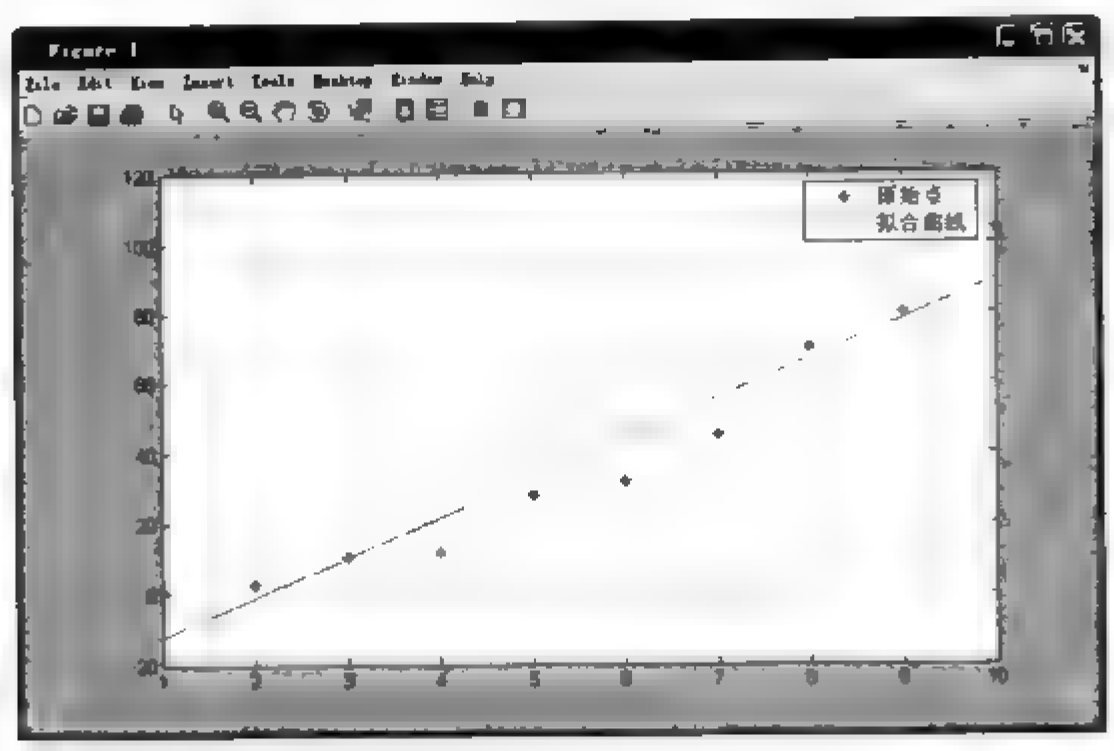


图 16-2　线性回归拟合曲线

从图 16-2 可以看出，拟合曲线和原来的数据点相比，相差比较大，精度不是很好，为此，用户可以增加 N 的值，例如用户选择 N=2 作为阶数，就会得到一个 2 阶多项式。继续在命令窗口中输入如下命令，并按 Enter 键确认，如图 16-3 所示。

```
>> b=polyfit(x,y,2)
b =

     1.1515    -1.3697     1.8000
>>
>>x2=0:0.05:10;
>> y2=b(1)*x2.^2+b(2)*x2+b(3);
>> plot(x,y,'*',x2,y2,'-r')
>>
```

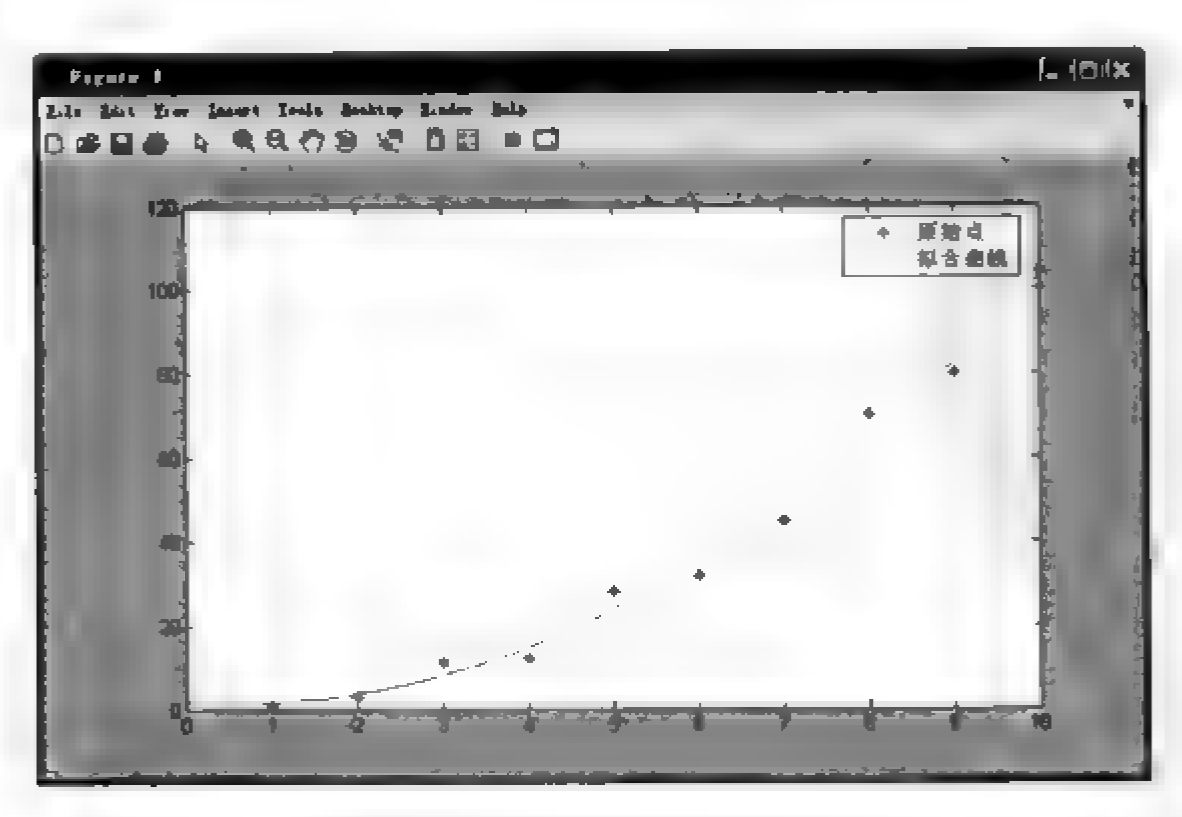


图 16-3　2 阶最小二乘多项式拟合

从图 16-3 可以看出，当 N 等于 2 时，所得拟合曲线与原数据拟合程度比较不错，那么是不是 N 值越大越好呢？由于多项式阶次的选择是有点任意的。两点决定一直线或一阶多项式。3 点决定一个平方或 2 阶多项式。依此类推，N+1 数据点惟一地确定 N 阶多项式。在本例中，有 10 个数据点，用户可以选择一个 9 阶的多项式。使得拟合曲线经过每一个试验数据点。

```
>> x9=-1:0.05:11;
>> c=polyfit(x,y,9)
Warning: Polynomial is badly conditioned. Remove repeated data points
         or try centering and scaling as described in HELP POLYFIT.
> In polyfit at 81
ans =
  1.0e+003 *
  Columns 1 through 9
-0.0000    0.0001   -0.0023    0.0289   -0.2191    1.0468   -3.1230    5.5447   -5.2321
  Column 10
    1.9570
>>
```

可见，由于拟合的阶数太高，MATLAB 7.0 自动给出警告信息，提醒用户太高的阶数会产生不太好的结果。为了比较拟合曲线和原始数据的拟合程度，用户可以绘制它们的图形，继续在命令窗口中输入如下命令，并按 Enter 键确认，如图 16-4 所示。

```
>> x9=1:0.05:10;
>>y9=c(1)*x9.^9+c(2)*x9.^8+c(3)*x9.^7+c(4)*x9.^6+c(5)*x9.^5+...
   c(6)*x9.^4+c(7)*x9.^3+c(8)*x9.^2+c(9)*x9+c(10);
>> plot(x,y,'*',x9,y9,'-r')
>>
```

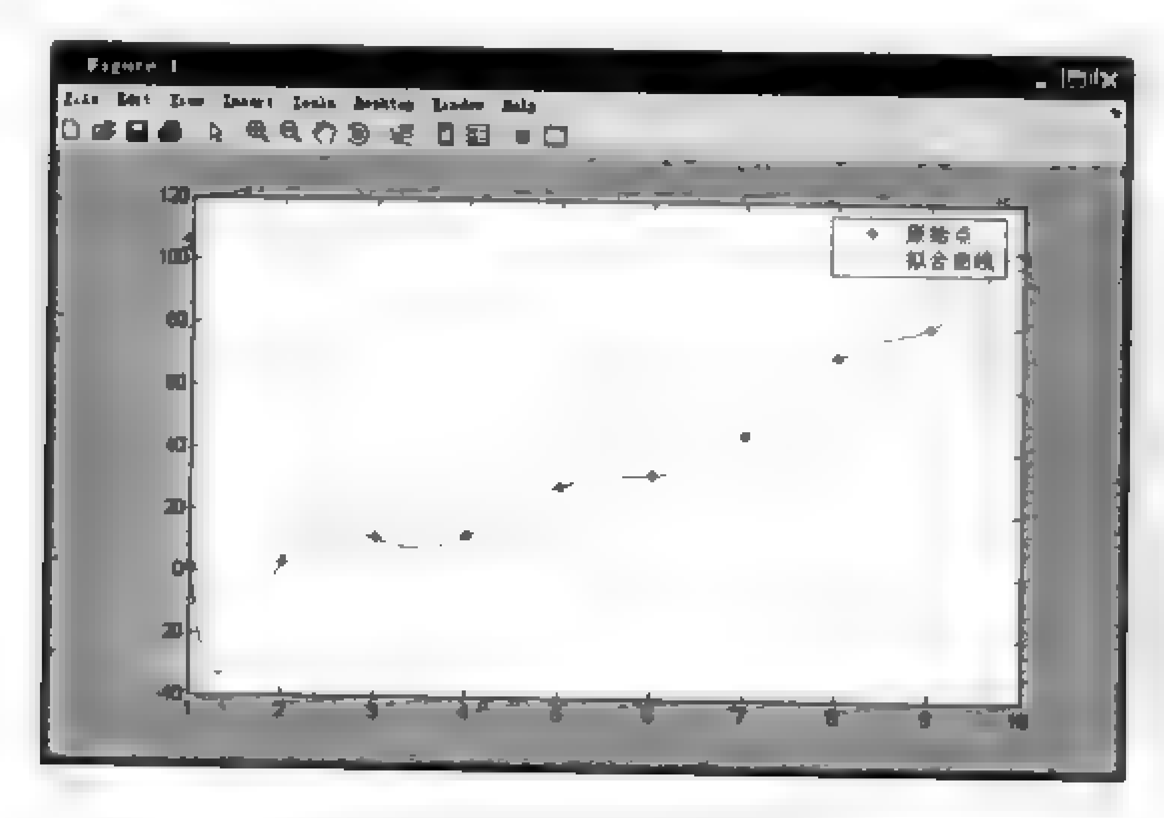


图 16-4　9 阶最小二乘多项式拟合

此时，用户可以看出，虽然所得的拟合曲线经过每一个原始数据点，但是总体的拟合曲线并不好。在 9 阶拟合中，在左边和右边的极值处，数据点之间出现大的纹波。如要进行高阶曲线拟合，这种纹波现象会经常发生，因此“阶数越多就越好”的观念在这里不适用。

16.2　曲 线 插 值

在实际工作中，人们得到的一些数据通常是一些不连续的点，在土木工程、流体力学、经济学和空气动力学等学科中经常要遇到这样的问题。此时，这些数据如果不加以处理，就难以发现内在的规律性。比如，用户要想得到这些分散点外的其他数值，就必须运用这些已知的点进行插值。本节将具体介绍这方面的内容。

16.2.1　拉格朗日插值

拉格朗日插值是一种普遍的应用比较插值方法，它要求在插值节点上，插值函数和原函数的值相等。它的基本原理为：X 为给定的 N 个插值节点，Y 为对应的 N 个函数值，利用 N 次拉格朗日插值多项式，可以求出插值区间内的任意 x 的插值。

N 次拉格朗日插值在 x 点的插值公式为：

$$y(x)=\sum_{k=1}^{N}Y_k\left(\prod_{\substack{j=1\\j\neq k}}^{N}\frac{X-X_j}{X_k-X_j}\right)$$

例 16-2　拉格朗日插值的实现。

解：编制 M 文件如下。

```
%该方法用于求解 lagrange 插值
%使用格式为 y=lagrange(X,Y,x)
%其中 X 为插值节点，Y 为插值节点的函数值，x 为所求点
function y=lagrange(X,Y,x)
n=length(X);
m=length(x);
for i=1:m
    z=x(i);
    s=0.0;
    for k=1:n
        p=1.0;
        for j=1:n
            if j~=k
                p=p*(z-X(j))/(X(k)-X(j));
            end
```

```
            end
            s=p*Y(k)+s;
        end
        y(i)=s;
    end
```

将该程序以文件名 lagrange.m 保存，在命令窗口中运行该程序如下。

```
>>X=2:0.5:10;
>>Y=log(X);
>> lagrange(X,Y,5.12)
ans =
      1.6332
>>
>> lagrange(X,Y,15)
ans =
   -51.9765
>>
```

稍微分析一下，发现在 x=5.12 时的插值比较合理，而 x=15 时的插值就与实际不相符合，这是因为 x=15 已经不在向量 X 的区域之内。为了比较插值函数与原函数的值，可以编写程序绘制图形如下。

```
>> x1=2:0.5:13;
>> y1=lagrange(X,Y,x1);
>> plot(X,Y,'b-',x1,y1,'r*')
>>
```

所绘制的图形如图 16-5 所示，可见，使用拉格朗日插值时，在插值向量 X 区域内的插值一般比较准确，但是在向量 X 区域外则不太准确，甚至差别比较大。如从图 16-5 中可以看出，当 x>12 时，插值曲线与原曲线相比，差别是相当大的。

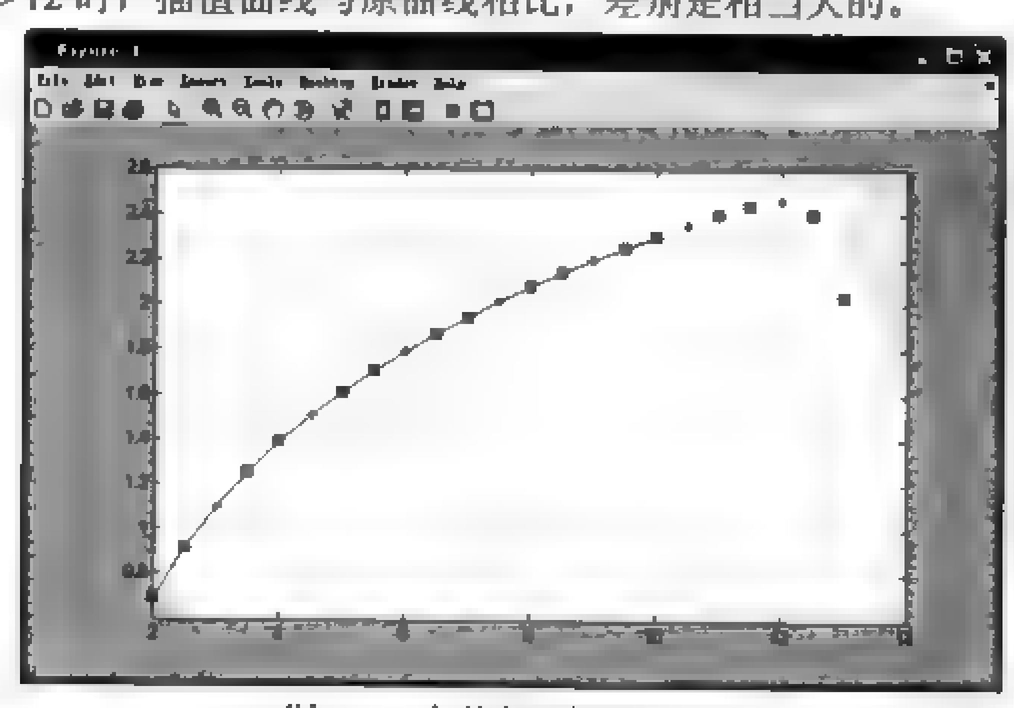


图 16-5　拉格朗日插值的使用

16.2.2 hermite 插值

与拉格朗日插值相比，hermite 插值是一种更为高级的插值方式，它不仅要求在插值节点上，插值函数和原函数的值相等，而且要求在节点上的导数值也相等。为了简单起见，下面只介绍一阶导数存在且相等的情况。

其基本原理如下。

X 为给定的 N 个插值节点，Y 为对应的 N 个函数值，Y1 为对于点的一阶导数值。利用 N 次 hermite 插值多项式，可以求出插值区间内的任意 x 的插值。

N 次 hermite 插值多项式在 x 点的插值公式如下。

$$y(x)=\sum_{i=1}^{N}h_i[(X_i-x)(2*a_i*Y_i-Y_i')+Y_i]$$

其中 $h_i=\prod_{\substack{j=1\\j\neq i}}^{N}(\frac{x-X_j}{X_i-X_j})^2$，$a_i=\sum_{\substack{i=1\\j\neq i}}^{N}\frac{1}{X_i-X_j}$

例 16-3 hermite 插值的实现。

解：编制 M 文件如下。

```
%该方法用于求解 hermite 插值
%使用格式为 y=ermite(X,Y,y1,x)
%其中 X 为插值节点，Y 为插值节点的函数值，y1 为在该点的一阶导数，x 为所求点
function y=hermite(X,Y,y1,x)
n=length(X);
m=length(x);
for i=1:m
    y0=0;
    for j=1:n
        h=1;
        a=0;
        for k=1:n
            if k~=j
                h=h*((x(i)-X(k))/(X(j)-X(k)))^2;
                a=1/(X(j)-X(k))+a;
            end
        end
        y0=y0+h*(((X(j)-x(i))*(2*a*Y(j)-y1(j))+Y(j)));
    end
        y(i)=y0;
end
```

将该程序以文件名 hermite.m 保存，在命令窗口中运行该程序如下。

```
>> X=[1:0.5:10];
>> Y=sin(X)+log(X)
Y =
  Columns 1 through 10
0.8415    1.4030    1.6024    1.5148    1.2397    0.9020    0.6295    0.5265    0.6505
0.9992
  Columns 11 through 19
1.5123    2.0869    2.6029    2.9529    3.0688    2.9386    2.6093    2.1761    1.7586
>>   x=[1:0.5:10];
>> Y1=cos(X)+1./X
>> y=hermite(X,Y,Y1,x)
y =
  Columns 1 through 10
0.8415    1.4030    1.6024    1.5148    1.2397    0.9020    0.6295    0.5265    0.6505
0.9992
  Columns 11 through 19
1.5123    2.0869    2.6029    2.9529    3.0688    2.9386    2.6093    2.1761    1.7586
>> plot(X,Y,'b-',x,y,'r*')
>>
```

从上边的数据可以看出，对于本例来说，在向量 X 的区域内插值是精确的。绘制的插值曲线和原曲线的对比如图 16-6 所示。

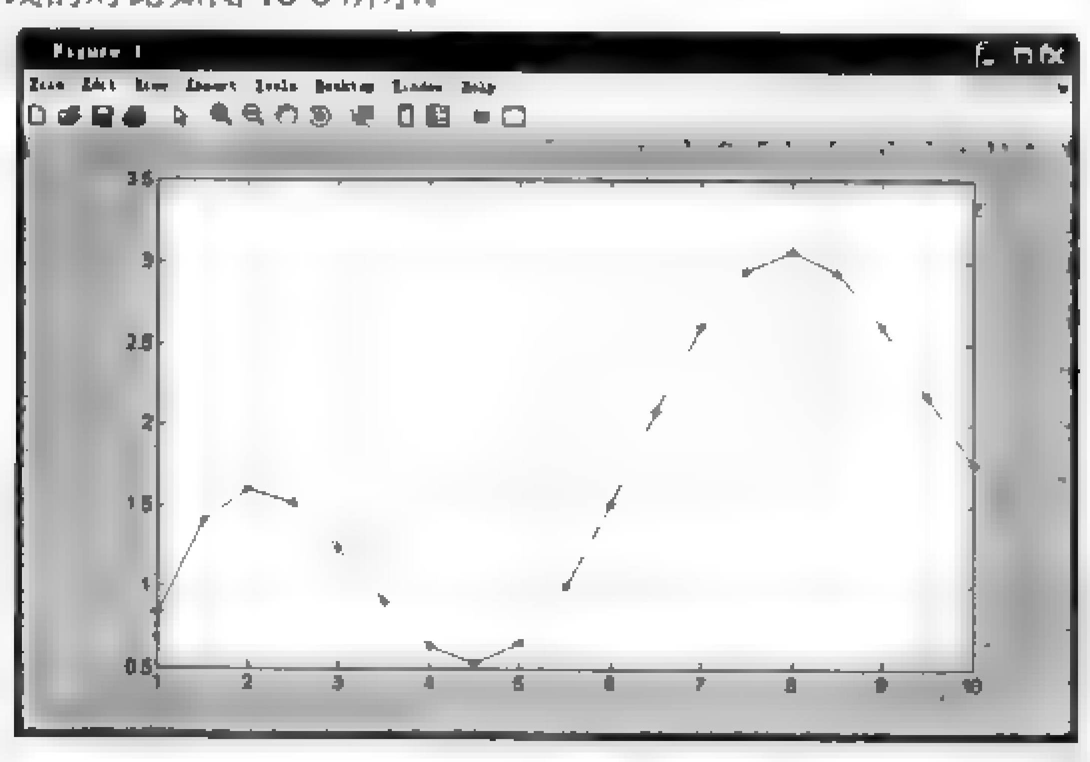


图 16-6　hermite 插值曲线 1

为了检验在向量 X 区域外的插值情况，可以在命令窗口中输入程序如下。

```
>> x=1:0.5:11.5;
>> y=hermite(X,Y,Y1,x);
>> YY=sin(x)+log(x);
>> plot(x,YY,'b-',x,y,'r*')
>>
```

所得插值图形如图 16-7 所示。

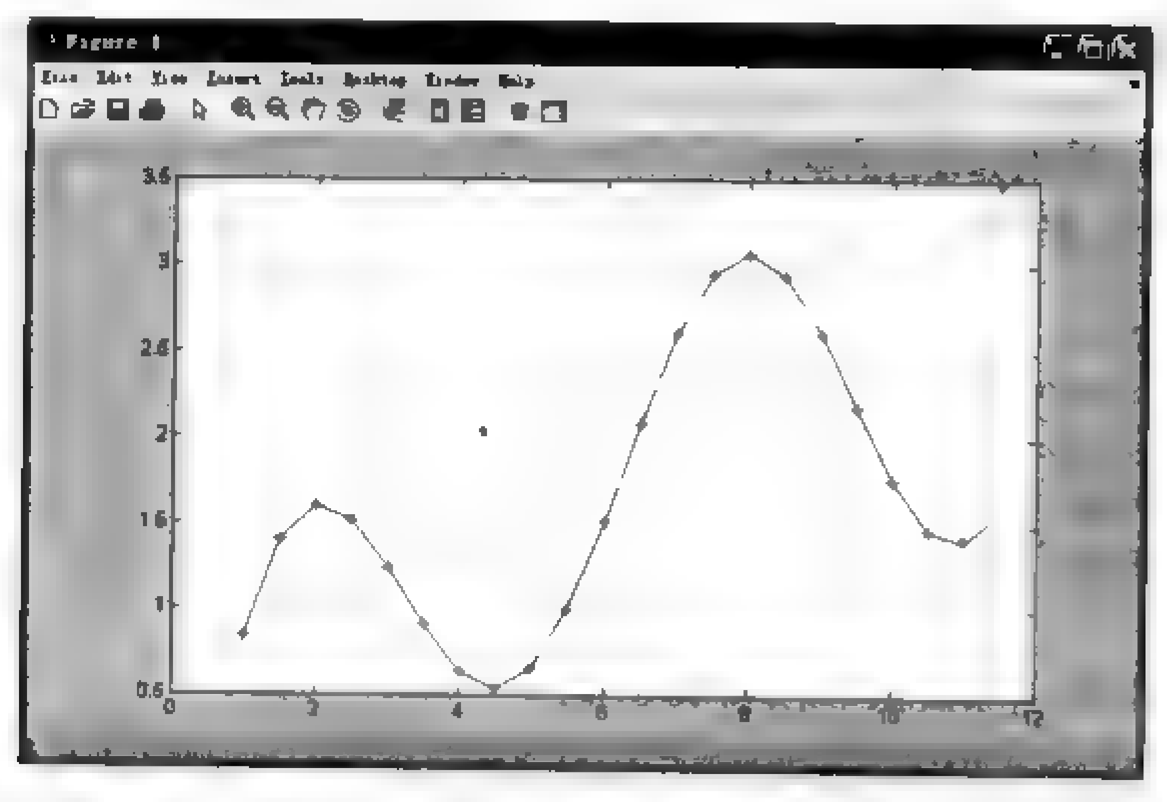


图 16-7　hermite 插值曲线 2

16.2.3　三次样条插值

由于使用高阶多项式的插值往往会产生病态的结果，因此有必要找出消除这种病态的方法。目前已经有多种方法。在这些方法中，三次样条插值是最常用的一种。在 MATLAB 7.0 中，实现基本的三次样条插值的函数有 spline、ppval、mkpp 和 unmkpp。

在三次样条中，要寻找三次多项式，以逼近每对数据点间的曲线。在样条术语中，这些数据点称之为断点。因为，两点只能决定一条直线，而在两点间的曲线可用无限多的三次多项式近似。因此，为使结果具有惟一性。在三次样条中，增加了三次多项式的约束条件。通过限定每个三次多项式的一阶和二阶导数，使其在断点处相等，就可以较好地确定所有内部三次多项式。此外，近似多项式通过这些断点的斜率和曲率是连续的。然而，第一个和最后一个三次多项式在第一个和最后一个断点以外，没有伴随多项式。因此必须通过其他方法确定其余的约束。最常用的方法，也是函数 spline 所采用的方法，就是采用非扭结条件。这个条件强迫第一个和第二个三次多项式的三阶导数相等。对最后一个和倒数第二个三次多项式也做同样地处理。

基于上述描述，人们可能猜想到，寻找三次样条多项式需要求解大量的线性方程。实际上，给定 N 个断点，就要寻找 N-1 个三次多项式，每个多项式有 4 个未知系数。这样，所求解的方程组包含有 4*(N-1)个未知数。把每个三次多项式列成特殊形式，并且运用各种约束，通过求解 N 个具有 N 个未知系数的方程组，就能确定三次多项式。这样，如果有 50 个断点，就有 50 个具有 50 个未知系数的方程组。幸好，用稀疏矩阵，这些方程式能够简明地列出并求解，这就是函数 spline 所使用的计算未知系数的方法。

(1) 使用 spline 函数做三次样条插值。

- $YY = spline(X,Y,XX)$ 命令使用三次样条法求出 YY，YY 是隐函数 Y 在向量

XX 所在点的值。X 和 Y 是已知的数值点。如果 Y 是一个矩阵，那么该命令将对 Y 的每一列进行计算。

- $pp = spline(X,Y)$ 命令返回精确的三次样条插值多项式。

例 16-4　三次样条插值的使用。

解：首先使用 $YY = spline(X,Y,XX)$ 命令如下。

```
>>x=0 : 12;
>>y=tan(pi*x/25);
>>xi=linspace(0, 12);
>>yi=spline(x, y, xi) ;
>>plot(x, y, ' o ', xi, yi), title(' Spline fit ')
>>
```

所绘制的三次样条插值曲线如图 16-8 所示。

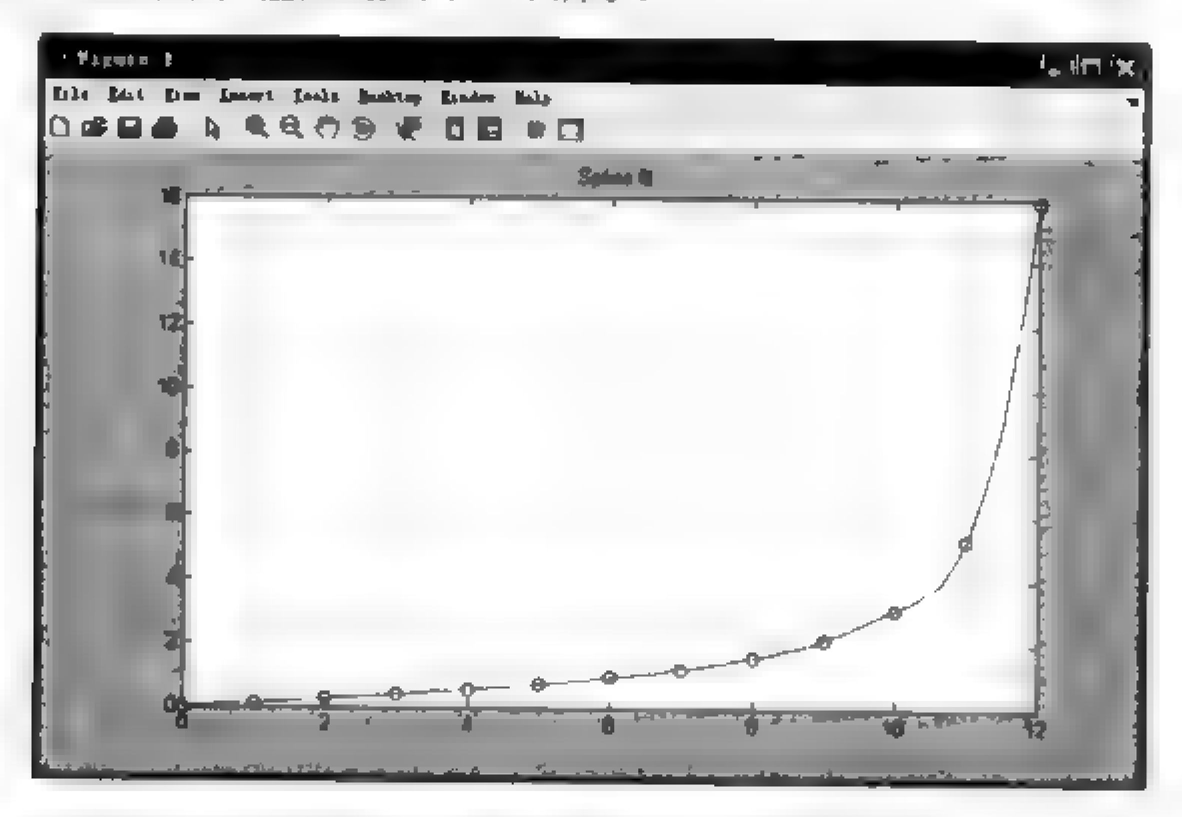


图 16-8　三次样条插值曲线 1

该方法适合于只需要一组内插值的情况。不过，如果需要从相同数据集里获取另一组内插值，再次计算三次样条系数是没有意义的。在这种情况下，可以调用仅带前两个参量的 $pp = spline(X,Y)$。继续在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> x = -4:4;
>>y = [0 .15 1.12 2.36 2.36 1.46 .49 .06 0];
>>cs = spline(x,[0 y 0]);
>>xx = linspace(-4,4,101);
>>plot(x,y,'o',xx,ppval(cs,xx),'-');
>>
```

该程序所绘制的插值曲线如图 16-9 所示。

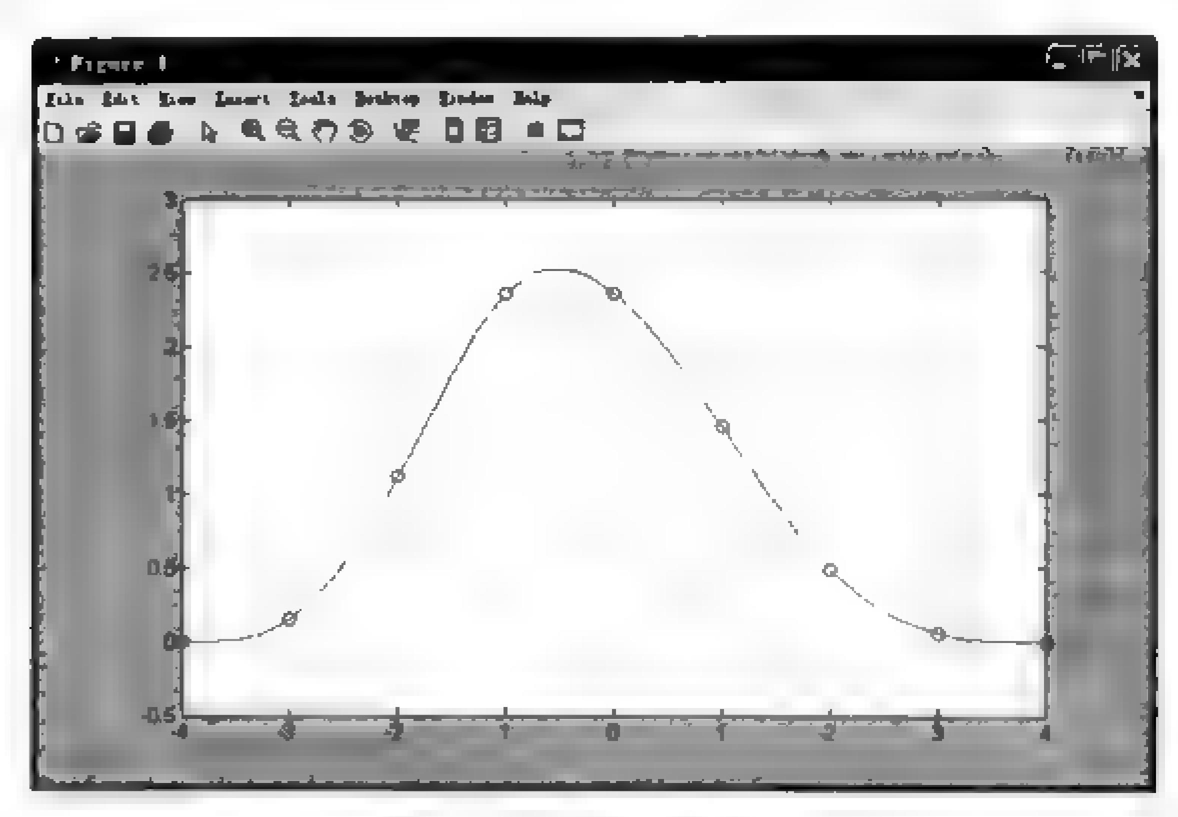


图 16-9　三次样条插值曲线 2

当采用该命令调用时，spline 函数返回一个称之为三次样条的 PP 形式或分段多项式形式的数组。这个数组包含了对于任意一组所期望的内插值和计算三次样条所必须的全部信息。给定 PP 形式，函数 ppval 计算该三次样条。

```
>>yi=ppval(pp, xi);    %计算先前计算过的同样的 yi
>>xi2=linspace(10, 12);
>>yi2=ppval(pp, xi2); %运用 pp 形式，在限定的更细区间[10，12]内，再次计算该三次样条
>>xi3=10 : 15
>>yi3=ppval(pp, xi3)
yi3 =
    3.0777    5.2422    15.8945    44.0038    98.5389   188.4689
```

它表明，可在计算三次多项式所覆盖的区间外，计算三次样条。当数据出现在最后一个断点之后或第一个断点之前时，则分别运用最后一个或第一个三次多项式来寻找内插值。

上述给定的三次样条的 PP 形式，存储了断点和多项式系数，以及关于三次样条表示的其他信息。因为，所有信息都被存储在单个向量里，所以这种形式在 MATLAB 7.0 中是一种方便的数据结构。当要计算三次样条表示时，必须把 PP 形式分解成它的各个表示段。在 MATLAB 7.0 中，通过函数 unmkpp 完成这一过程。运用上述 PP 形式，该函数给出如下结果。

```
>>[break,  coefs,  npolys,  ncoefs]=unmkpp(pp)
breaks =
  Columns 1 through 12
     0     1     2     3     4     5     6     7     8     9    10    11
  Column 13
    12
coefs =
```

```
    0.0007   -0.0001    0.1257         0
    0.0007    0.0020    0.1276    0.1263
    0.0010    0.0042    0.1339    0.2568
    0.0012    0.0072    0.1454    0.3959
    0.0024    0.0109    0.1635    0.5498
    0.0019    0.0181    0.1925    0.7265
    0.0116    0.0237    0.2344    0.9391
   -0.0083    0.0586    0.3167    1.2088
    0.1068    0.0336    0.4089    1.5757
   -0.1982    0.3542    0.7967    2.1251
    1.4948   -0.2406    0.9102    3.0777
    1.4948    4.2439    4.9136    5.2422
npolys =
    12
ncoefs =
     4
```

这里 break 是断点，coefs 是矩阵，它的第 i 行是第 i 个三次多项式，npolys 是多项式的数目，ncoefs 是每个多项式系数的数目。注意，这种形式非常一般，样条多项式不必是三次。这对于样条的积分和微分是有用的。

给定上述分散形式，函数 mkpp 恢复了 PP 形式。

```
>>pp=mkpp(break, coefs)
pp =
  Columns 1 through 7
   10.0000    1.0000   12.0000         0    1.0000    2.0000    3.0000
  Columns 8 through 14
    4.0000    5.0000    6.0000    7.0000    8.0000    9.0000   10.0000
  Columns 15 through 21
   11.0000   12.0000    4.0000    0.0007    0.0007    0.0010    0.0012
  Columns 22 through 28
    0.0024    0.0019    0.0116   -0.0083    0.1068   -0.1982    1.4948
  Columns 29 through 35
    1.4948   -0.0001    0.0020    0.0042    0.0072    0.0109    0.0181
  Columns 36 through 42
    0.0237    0.0586    0.0336    0.3542   -0.2406    4.2439    0.1257
  Columns 43 through 49
    0.1276    0.1339    0.1454    0.1635    0.1925    0.2344    0.3167
  Columns 50 through 56
    0.4089    0.7967    0.9102    4.9136         0    0.1263    0.2568
  Columns 57 through 63
    0.3959    0.5498    0.7265    0.9391    1.2088    1.5757    2.1251
  Columns 64 through 65
    3.0777    5.2422
```

16.3 习　　题

1. 设有如下数据表：

x	0	0.5	1	1.5	2	2.5	3	3.5
y	1	2.4	3.1	5.0	7	11	17	24

试采用最小二乘法对上述表格所提供的数据进行拟合。

2. 设有如下数据表：

x	0	0.5	1	1.5	2	2.5	3	3.5
$y=e^x$	1.0000	1.6487	2.7183	4.4817	7.3891	12.1825	20.0855	33.1155

试采用拉格朗日插值法求当 x 分别取 0.64、1.67、2.04 和 3.35 时函数 Y 的近似值。并与实际值进行比较。

3. 使用 MATLAB 7.0 的帮助系统学习 interp1 函数的使用方法，该函数可以实现多种形式的插值，包括线性最佳值插值、线性插值、三次样条插值和三次插值，试使用该函数重新计算习题 16.1 和习题 16.2。

4. 设有如下数据表：

x	0.1	0.2	0.3	0.4	0.5
$y=\sin(x)$	0.0998	0.1987	0.2955	0.3894	0.4794
$y'=\cos(x)$	0.9950	0.9801	0.9553	0.9211	0.8776

试使用 hermite 插值法求 x 在区间[0.1,0.5]之间的值，以 0.02 为步长，然后绘制插值曲线，并将其与原曲线对比。

5. 使用三次样条法重新计算习题 16.4。

第17章　普通方程和微分方程

本章将主要介绍线性方程组、非线性方程组、常微分方程和偏微分方程的求解。

17.1　方程组的求解

在科学研究和工程实践中，很多情况下都需要求解各种方程组。一般来说，方程组有线性和非线性两种，本节将介绍在 MATLAB 7.0 中线性和非线性方程组的解法。

17.1.1　线性方程组的解法

对于一般的矩阵，可以采用直接法和迭代法两种方法求解线性方程组，而对于大型的稀疏矩阵，采用直接法则耗时太多，此时，采用迭代法是一种不错的选择。

1. 直接法

(1) 使用“\”或“/”两种符号直接求解线性方程组

在 MATLAB 7.0 语言中求解线性方程组的最简单的办法是矩阵除法，它只需使用简单的“\”或“/”这两种符号就可以实现。下面举一例子予以说明。

例 17-1　使用“\”或“/”两种符号直接求解线性方程组。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> a=magic(5)
a =
    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9
>> b=diag(ones(5))
b =
     1
     1
     1
     1
     1
>> a\b
```

```
ans =
      0.0154
      0.0154
      0.0154
      0.0154
      0.0154
```

(2) 使用 lu 分解来直接求解线性方程组

在 MATLAB 7.0 语言中提供了 lu 函数来进行矩阵的 lu 分解，使用它用户可以很方便地求解线性方程组。其使用格式如下。

- $[l,u]=lu(x)$ 命令产生一个上三角阵 u 和一个下三角阵 l，值得注意的是 x 不一定为方阵。
- $[l,u,p]=lu(x)$ 命令返回一个下三角阵 l 和一个上三角阵 u，并返回一个交换矩阵 p，使得 p 满足 $p*x=l*u$。
- 当 x 是非稀疏矩阵时，$y=lu(x)$ 命令返回 LAPACK'S DGETRF 或 ZGETRF 的结果；当 x 是稀疏矩阵时，$y=lu(x)$ 命令返回一个和上三角阵 u 镶嵌在一起的下三角阵 l。在这两种情况下，交换矩阵 p 都将不存在。
- 对稀疏且非空的矩阵 x，$[l,u,p,q]=lu(x)$ 命令返回一个下三角阵 l 和一个上三角阵 u，并返回一个交换矩阵 p 和一列记录矩阵 q。使得 $p*x*q=l*u$。

例 17-2　使用 lu 分解直接求解线性方程组。

解：求解方程 $X*y=b$，X 和 b 的定义见程序。

```
>> X=[3 7 7;1 7 0;2 3 5]
X
     3     7     7
     1     7     0
     2     3     5
>> [L,U] = lu(X)
L =

    1.0000         0         0
    0.3333    1.0000         0
    0.6667   -0.3571    1.0000
U
    3.0000    7.0000    7.0000
         0    4.6667   -2.3333
         0         0   -0.5000
>> b=[1 2 3]'
b =

     1
     2
```

```
        3
>> Y1=L\b
Y1 =
    1.0000
    1.6667
    2.9286
>> y=U\Y1
y =
   20.0000
   -2.5714
   -5.8571
>>
```

2. 迭代法

迭代法主要有 Jacobi 迭代、Gauss-Seidel 迭代和 SOR 迭代，它们各有特点，下面分别予以介绍。

(1) Jacobi 迭代法

该方法的基本思路如下。

设方程组 $Ax=b$，其中 A 为 N 阶方阵，b 为 N 阶列向量，且 A 非奇异，显然 A 可以分解为 $A=D-L-U$。此时，有 $x=D^{-1}*(L+U)*x+D^{-1}*b$，由此可以构造迭代算法 $x^{k+1}=B*x^k+f$，其中，$B=D^{-1}*(L+U)=I-D^{-1}*A$，$f=D^{-1}*b$。

例 17-3　Jacobi 迭代法的 M 文件。

解：编写 M 文件如下。

```
%该函数用 Jacobi 迭代法求解线性方程组
%用户需要输入 3 个参数
%设 A×y=b，用户输入矩阵 A 和 b
%再输入一个初始向量 x0
%使用格式为 y=jacobi(A,b,x0)
function y=jacobi(A,b,x0)
D=diag(diag(A));
U=-triu(A,1);
L=-tril(A,-1);
B=-D\(L+U);
f=D\b;
y=B*x0+f;
n=1;
while norm(y-x0)>=1.0e-6&n<=1000
     x0=y;
     y=B*x0+f;
     n=n+1;
end
fprintf('方程组的解 y=');
```

```
y
fprintf('\n'),
fprintf('迭代次数 n=',y);
n
```

例 17-4　使用 Jacobi 迭代法求解线性方程组。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> A=[10 3 1;2 -10 3;1 3 10]
A =
    10     3     1
     2   -10     3
     1     3    10
>> b=[14 -5 14]'
b =
    14
    -5
    14
>> x0=[0 0 0]'
x0 =
     0
     0
     0
>> y=jacobi(A,b,x0)
方程组的解 y=
y =
    1.0000
    1.0000
    1.0000
迭代次数 n=
n =
    17
y =
    1.0000
    1.0000
    1.0000
>>
```

(2) Gauss-Seidel 迭代法

该方法的基本思路如下。

设方程组 $Ax=b$，其中 A 为 N 阶方阵，b 为 N 阶列向量，且 A 非奇异，显然 A 可以分解为 $A=D-L-U$。此时，有 $x=(D-L)^{-1}*U*x+(D-L)^{-1}*b$，由此可以构造迭代算法 $x^{k+1}=G*x^k+f$，其中，$G=(D-L)^{-1}*U$，$f=(D-L)^{-1}*b$。

例 17-5　Gauss-Seidel 迭代法的 M 文件。

解：编写 M 文件如下。

```
%该函数用 Gauss-Seidel 迭代法求解线性方程组
%用户需要输入 3 个参数
%设 A×y=b，用户输入矩阵 A 和 b
%再输入一个初始向量 x0
%使用格式为 y= guassseidel (A,b,x0)
function y=gaussseidel(A,b,x0)
D=diag(diag(A));
U=-triu(A,1);
L=-tril(A,-1);
G=(D-L)\U;
f=(D-L)\b;
y=G*x0+f;
n=1;
while norm(y-x0)>=1.0e6&n<=1000
      x0=y;
      y=G*x0+f;
      n=n+1;
end
fprintf('方程组的解 y=');
y
fprintf('\n');
fprintf('迭代次数 n=',y);
n
```

例 17-6　使用 Gauss-Seidel 迭代法求解线性方程组。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> A=[10 3 1;2 -10 3;1 3 10]
A =
      10      3      1
       2    -10      3
       1      3     10
>> b=[14 -5 14]'
b
      14
      -5
      14
>> x0=[0 0 0]'
x0 =
       0
       0
       0
```

```
>> gaussseidel(A,b,x0)
方程组的解 y=
    y
        1.0000
        1.0000
        1.0000
    迭代次数 n=
    n =
        10
    ans =
        1.0000
        1.0000
        1.0000
    >>
```

比较上边的两种迭代法，可以看出 Gauss-Seidel 迭代法比 Jacobi 迭代法的收敛速度要更快一些，但是，在有些情况下，可能会出现使用 Jacobi 迭代法收敛而使用 Gauss-Seidel 迭代法不收敛的情况。

(3) SOR 迭代法

由于 Jacobi 迭代法和 Gauss-Seidel 迭代法的收敛速度一般比较慢，于是又产生了一种新的方法，那就是逐次超松弛迭代法，简称 SOR 法。其格式如下：

设方程组 $Ax=b$，其中 A 为 N 阶方阵，b 为 N 阶列向量，且 A 非奇异，显然 A 可以分解为 $A=D-L-U$。由此可以构造迭代算法 $x^{k+1}=w*L*x^k+w*(D-w*L)-1*b$。其中，w 为松弛因子，它的最佳取值在[1,2]之间。

例 17-7　SOR 迭代法的 M 文件。

解：编写 M 文件如下。

```
%该函数用 SOR 迭代法求解线性方程组
%用户需要输入 4 个参数
%设 A×y=b，用户输入矩阵 A 和 b
%再输入一个初始向量 x0 和松弛稀疏 w
%使用格式为 y= guassseidel (A,b,w,x0)
function y=sor(A,b,w,x0)
D=diag(diag(A));
U=triu(A,1);
L=tril(A,-1);
lw=(D-w*L)\((1-w)*D+w*U);
f=(D-w*L)\b*w;
y=lw*x0+f;
n=1;
while norm(y-x0)>=1.0e-6&n<=1000
      x0=y;
      y=lw*x0+f;
```

```
n=n+1;
end
fprintf('方程组的解 y=');
y
fprintf('\n');
fprintf('迭代次数 n=',y);
n
```

例 17-8　使用 SOR 迭代法求解线性方程组。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> A=[10 3 1;2 -10 3;1 3 10]
A =
    10     3     1
     2   -10     3
     1     3    10
>> b=[14 -5 14]'
b =
    14
    -5
    14
>> x0=[0 0 0]'
x0 =
     0
     0
     0
>> sor(A,b,1.2,x0)
方程组的解 y=
y =
    1.4762
   -0.2381
    1.4762
迭代次数 n=
n =
    22
ans =
    1.4762
   -0.2381
    1.4762
>> sor(A,b,1.1,x0)
方程组的解 y=
y =
    1.4762
   -0.2381
    1.4762
```

```
迭代次数 n=
n =
      14
ans =
      1.4762
     -0.2381
      1.4762
>> sor(A,b,1.01,x0)
方程组的解 y=
y =
      1.4762
     -0.2381
      1.4762
迭代次数 n=
n =
       9
ans =
      1.4762
     -0.2381
      1.4762
>>
```

从程序的运行结果可以看出，采用不同的 w 值，对收敛速度会产生很大的影响。

17.1.2 非线性方程组的解法

对于非线性方程组的求解，一般可以采用迭代法、二分法和遍历法来求解。由于最后那一种方法效率不高，实际工作中很少使用。MATLAB 7.0 语言提供了 fsolve 函数来求解非线性方程组，下面将对其予以具体介绍。

在 MATLAB 7.0 语言中，使用 fsolve 函数来求解非线性方程组 $f(x)=0$。其中 f 和 x 可以是向量或矩阵。其使用方法如下。

- $x=fsolve(fun,x0)$ 命令以 $x0$ 为初始矩阵来求解方程 fun，fun 接受输入量 x 并返回一个向量(矩阵)，使得 $f=fun(x)$。
- $x=fsolve(fun,x0,options)$ 命令以 $options$ 为选择参数的输入变量，详细内容用户可以查看 optimset 的帮助信息得知，
- $x=fsolve(fun,x0,options,p1,p2,\ldots)$ 命令将问题定性参数 $p1$、$p2$ 和 $p3$ 等直接赋值给函数 $fun(x,p1,p2,p3,\ldots)$。而当 $options$ 为默认值时，该命令将返回一个空矩阵。
- $[x,feval]=fsolve(fun,x0,\ldots)$ 命令返回客观方程在 x 处的值。
- $[x,feval,exitflag]=fsolve(fun,x0,\ldots)$ 命令返回一个描述 $fsolve$ 的溢出情况的字符串 $exitflag$。

当 $exitflag > 0$ 时， $fsolve$ 的解将收敛到 x 。

当 $exitflag = 0$ 时，将取得方程的最大解数。

当 $exitflag < 0$ 时， $fsolve$ 的解在 x 处不收敛。

- $[x, feval, exitflag, output, jacob] = fsolve(fun, x0, ...)$ 命令返回函数 fun 在 x 处的 Jacobian 解。

例 17-9　使用 fsolve 函数求解非线性方程组。

解：首先创建定义 FUN 方程组的 M 文件如下，并将函数命名为 feixianxing.m。

```
function f=feixianxing(x)
a=x(1);
b=x(2);
c=x(3);
f(1)=a^2+b+sin(c);
f(2)=a*b+c;
f(3)=cos(a)+b^2+2*c;
end
```

继续在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> x0=[1 1 1]
x0 =
     1     1     1
>> f=fsolve('feixianxing',x0)
f =
    0.5270    0.2309   -0.4113
>>
```

17.2　微分方程的求解

在现在科学研究和工程实践中，很多数学模型都是用微分方程来确定的，很多基本方程本身就是一个微分方程。因此，求解常微分方程和偏微分方程是非常重要的，但是，大部分的微分方程目前还难以得到其解析解，因此，人们只有利用计算机强大的计算功能来求其数值解，MATLAB 7.0 语言提供了求解这两种问题的强大功能，本节将对其进行详细介绍。

17.2.1　常微分方程的数值求解

在 MATLAB 7.0 语言中，提供了一些专门用来求解常微分方程的功能函数，主要有 ode23、ode45、ode113、ode 15s、ode 23s、ode 23t 和 ode 23tb。下面将对其分别予以介绍。

首先介绍一些与上述功能函数配套的函数。

ode solvers：ode 23、ode 45、ode 113、ode 15s、ode 23S、ode 23t 和 ode 23tb。

参数选择函数：odeset、odeget。

输出函数：odeplot、odephas2、odephas3 和 odeprint。

结果评估函数：deval。

ode 示例：rigidode、ballode 和 orbitode。

MATLAB 7.0 提供的这些 ode 函数主要是采用 Runge-Kutta 法求解常微分方程，但是，在用不同阶数的 Runge-Kutta 法求解常微分方程时，会花费不同的时间，并产生不同精度的结果。表 17-1 就是对上边各个 ode 函数的简要介绍。

表 17-1　ode 函数的简要介绍

函 数 名	函 数 含 义	函 数 名	函 数 含 义
ode23	普通 2-3 阶法解 ode	ode23s	低阶法解刚性 ode
ode45	普通 4-5 阶法解 ode	ode23t	解适度刚性 ode
ode15s	变阶法解刚性 ode	ode23tb	低阶法解刚性 ode
ode113	普通变阶法解 ode		

另外，odeset 函数用于创建和更改 Solver 选项，而 odeget 函数用于读取 Solver 的设置值。odeplot 函数用于输出 ode 的时间序列图；odephas2 函数用于输出 ode 的二维相平面图；odephas3 函数用于输出 ode 的三维相平面图；odeprint 在命令窗口输出结果。

下面对表 17-1 中的各个 ode 函数分别予以介绍。

1. 普通 2-3 阶法解 ode

在 MATLAB 7.0 语言中，函数 ode23 使用普通 2-3 阶 Runge-Kutta 法求解常微分方程。其使用方法如下。

- $[t, y] = ode23(odefun, tspan, y0)$ 命令返回一个列向量，其中 F 定义此微分方程的形式 $y' = f(t, y)$，$tspan = [t0, tfinal]$ 表示此微分方程的积分限是从 $t0$ 到 $tfinal$，当形如 $tspan = [t0, t1, ..tfinal]$ 时，它也可以是一些离散的点。
- $[t, y] = ode23(odefun, tspan, y0, options)$ 命令设置 $options$ 为积分参数，包括 'RelTol'(相对误差)和'AbsTol'(绝对误差)等。详细内容可以参见 odeset 的帮助信息 。
- $[t, y] = ode23(odefun, tspan, y0, options, p1, p2, ...)$ 命令中，参数 $p1$ 和 $p2$ 等可传递给函数 $odefun$，其形式为 $odefun(t, y, p1, p2, ...)$。
- $[t, y, te, ye, ie] = ode23(odefun, tspan, y0, options...)$ 命令调用函数时，必须设置 $options$ 中的事件属性为 on，输出向量 TE 为一列向量，代表自变量点，ye 的行为对应点上的解，ie 代表解的索引。

例 17-10　使用 ode23 函数求解常微分方程 $y' = -y + x^2 + 4*x + 1$。

解：首先创建函数 odeliyi 如下。

```
function f=odeliyi(x,y)
f=-y+x^2+4*x+1;
end
```

继续在命令窗口中输入如下命令，并按 Enter 键确认。

```
>> [x,y]=ode23('odeliyi',[1,4],1);
>> x'
ans =
  Columns 1 through 10
1.0000    1.0160    1.0960    1.3960    1.6960    1.9960    2.2960    2.5960    2.8960    3.1960
  Columns 11 through 13
    3.4960    3.7960    4.0000
>> y'
ans =
  Columns 1 through 10
   1.0000   1.0801   1.4848   3.0674   4.7691   6.6056   8.5886  10.7269  13.0269   15.4932
  Columns 11 through 13
    18.1295    20.9384    22.9483
>>
```

2. 普通 4-5 阶法解 ode

在 MATLAB 7.0 语言中，函数 ode45 使用普通 4-5 阶 Runge-Kutta 法求解常微分方程。其使用方法与 ode23 函数的使用方法基本相同，在此不再赘述。

例 17-11　使用 ode45 函数求解常微分方程。

解：在命令窗口中输入如下命令，并按 Enter 键确认。

```
%本例求解方程 y' = vdp1(t,y)。
>>[t,y]=ode45(@vdp1,[0 20],[2 0]);
>> plot(t,y(:,1));
>>
```

绘制出的图形如图 17-1 所示。

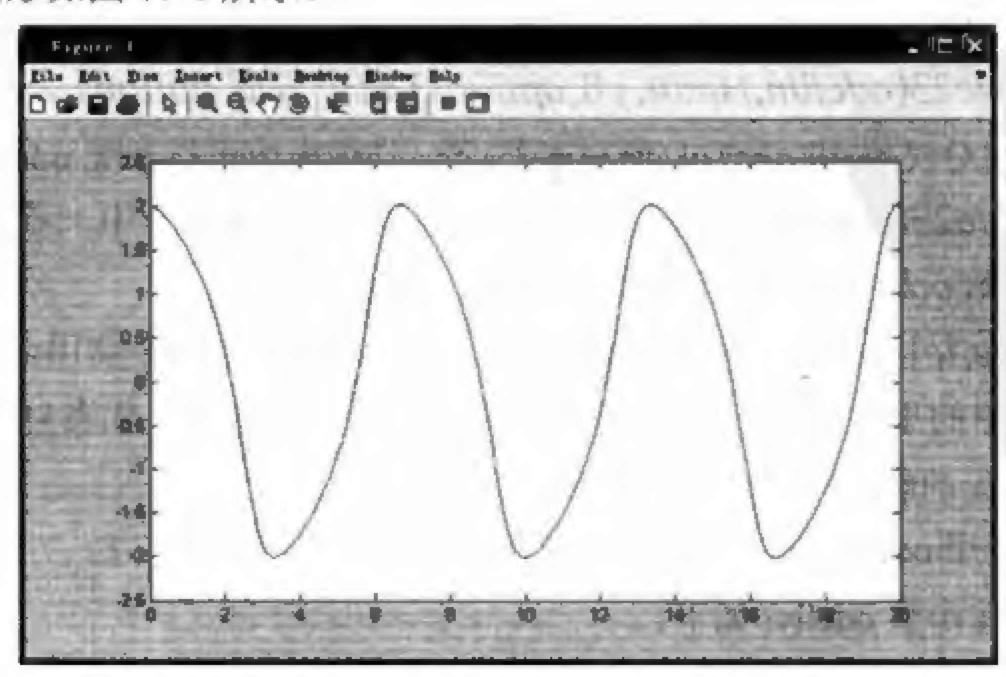


图 17-1　普通 4-5 阶法解 ode 所绘制图形

其他的几个函数使用方法和 ode 23 与 ode 45 大致相同，用户需要使用的时候，可以使用 MATLAB 7.0 的帮助系统详细了解。下面介绍一些各个函数的特点，用户可以根据需要选择使用，如表 17-2 所示。

表 17-2　不同求解器 Solver 的特点

求解器 Solver	ODE 类型	特　点	说　明
ode45	非刚性	一步算法；4、5 阶 Runge-Kutta 方程；累计截断误差达$(\Delta x)^3$	大部分场合的首选算法
ode23	非刚性	一步算法；2、3 阶 Runge-Kutta 方程；累计截断误差达$(\Delta x)^3$	使用于精度较低的情形
ode113	非刚性	多步法；Adams 算法；高低精度均可到 $10^{-3}\sim10^{-6}$	计算时间比 ode45 短
ode23t	适度刚性	采用梯形算法	适度刚性情形
ode15s	刚性	多步法；Gear's 反向数值微分；精度中等	若 ode45 失效时，可尝试使用
ode23s	刚性	一步法；2 阶 Rosebrock 算法；低精度	当精度较低时，计算时间比 ode15s 短
ode23tb	刚性	梯形算法；低精度	当精度较低时，计算时间比 ode15s 短

17.2.2　偏微分方程的数值求解

在 MATLAB 7.0 语言中，使用 pdepe 函数来求解偏微分方程，同时，使用 pdeval 函数来求取 pdepe 得到的解的值(或者对这个解进行差值)，下面将对它们的使用方法予以介绍。

- *sol = pdepe(m, pdefun, icfun, bcfun, xmesh, tspan)* 命令求解一维的抛物线——椭圆的初始边界值问题，该问题具有一个空间变量 x 和一个时间变量 t。
- pdeval 求解 pdepe 得到的解的值(或者对这个解进行差值)。

关于偏微分方程的解的进一步信息可以参见 MATLAB 7.0 的帮助系统。

例 17-12　使用 pdepe 函数求解偏微分方程。

解：在 MATLAB 7.0 的命令窗口中输入如下命令，并按 Enter 键确认。

```
>> x = linspace(0,1,20);
>> t = [0 0.5 1 1.5 2];
>> sol = pdepe(0,@pdex1pde,@pdex1ic,@pdex1bc,x,t);
>> u1 = sol(:,:,1);
>> surf(x,t,u1);
>>
```

该程序段绘制的图形如图 17-2 所示。

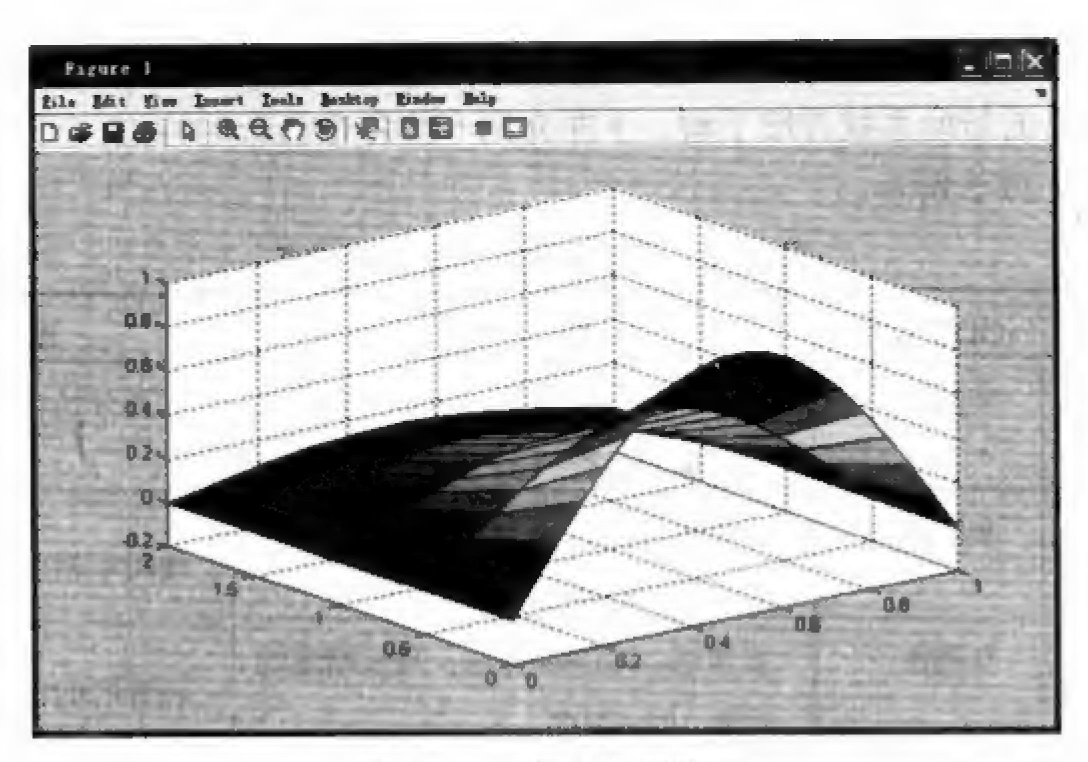


图 17-2　偏微分方程图形

17.3 习　题

1. 求解下列线性方程组。

(1) $$\begin{cases} 0.4x + 0.3124y + 2.6598z + 6.9785w = 0.24 \\ 3.142x + 8.22y + 6.16z + 0.254w = 3.251 \\ 0.1785x + 5.358y + 9.7932z + 3.846w = 0.21 \\ 2.643x + 8.321y + 0.283z + 6.735w = -2.354 \end{cases}$$

(2) $$\begin{cases} 0.9501x + 0.8913y + 0.8214z + 0.9218w = 0.24 \\ 0.2311x + 0.7621y + 0.4447z + 0.7382w = 0.5428 \\ 0.6068x + 0.4565y + 0.6154z + 0.1763w = 0.5376 \\ 0.4860x + 0.0185y + 0.7919z + 0.4057w = -0.5714 \end{cases}$$

2. 求解下列非线性方程组，要求误差在 0.01 之内。

(1) $x^2 - x - 1 = 0$

(2) $2x^3 + x^2 + 3x - 1 = 0$

(3) $$\begin{cases} x - 0.7\sin(x) - 0.2\cos(y) = 0 \\ y - 0.7\cos(x) + 0.2\sin(y) = 0 \end{cases}$$

3. 求解下列常微分方程。

(1) $y' = -2y + 3x^2 + 1$

(2) $y' = -y + 4x$